Counting Solutions to Polynomial Systems via Reductions*

Richard Ryan Williams

MIT CSAIL & EECS, Cambridge, MA, USA
rrw@mit.edu

Abstract

This paper provides both positive and negative results for counting solutions to systems of polynomial equations over a finite field. The general idea is to try to reduce the problem to counting solutions to a *single* polynomial, where the task is easier. In both cases, simple methods are utilized that we expect will have wider applicability (far beyond algebra).

First, we give an efficient deterministic reduction from approximate counting for a system of (arbitrary) polynomial equations to approximate counting for *one* equation, over any finite field. We apply this reduction to give a deterministic $\mathrm{poly}(n, s, \log p)/\varepsilon^2$-time algorithm for approximately counting the fraction of solutions to a system of s quadratic n-variate polynomials over $\mathbb{F}_p$ (the finite field of prime order p) to within an additive ε factor, for any prime p. Note that uniform random sampling would already require $\Omega(s/\varepsilon^2)$ time, so our algorithm behaves as a full derandomization of uniform sampling. The approximate-counting algorithm yields efficient approximate counting for other well-known problems, such as 2-SAT, NAE-3SAT, and 3-Coloring. As a corollary, there is a deterministic algorithm (with analogous running time) for producing solutions to such systems which have at least εp^n solutions.

Second, we consider the difficulty of exactly counting solutions to a single polynomial of constant degree, over a finite field. (Note that *finding* a solution in this case is easy.) It has been known for over 20 years that this counting problem is already NP-hard for degree-three polynomials over $\mathbb{F}_2$; however, all known reductions increased the number of variables by a considerable amount. We give a subexponential-time reduction from counting solutions to k-CNF formulas to counting solutions to a degree-$k^{O(k)}$ polynomial (over any finite field of $O(1)$ order) which *exactly preserves* the number of variables. As a corollary, the Strong Exponential Time Hypothesis (even its weak counting variant #SETH) implies that counting solutions to constant-degree polynomials (even over $\mathbb{F}_2$) requires essentially 2^n time. Similar results hold for counting orthogonal pairs of vectors over $\mathbb{F}_p$.

1998 ACM Subject Classification G.1.5 Numerical Analysis – Roots of Nonlinear Equations, I.1.2 Symbolic and Algebraic Manipulation – Algorithms

Keywords and phrases counting complexity, polynomial equations, finite field, derandomization, strong exponential time hypothesis

Digital Object Identifier 10.4230/OASIcs.SOSA.2018.6

1 Introduction

A canonical problem in pseudorandomness is:

> *Given a class $\mathcal{C}$ of Boolean circuits, is there a deterministic and efficient method for approximately counting the fraction of satisfying assignments to any circuit from $\mathcal{C}$?*

* Supported by NSF CAREER Grant CCF-1741615. A talk on an early version of this work can be found at https://youtu.be/gJxpUhc1Gfc

By uniformly random sampling $\Theta(1/\varepsilon^2)$ inputs to a given circuit, we can easily obtain an additive ε-approximation to the fraction of satisfying assignments, with high probability. Thus the above problem amounts to deterministically achieving what trivial random sampling can provide in a natural setting, with the target of reaching poly$(n)/\varepsilon^2$ time (or perhaps even better in some cases where randomness can also achieve more, such as approximately counting knapsack solutions [20]).

The general problem of finding such algorithms has been studied for decades. Of most relevance to this paper are the prior results for AC^0 circuits [2, 30, 8], for CNF/DNF of bounded or unbounded width [28, 23, 32, 21], and for $\mathbb{F}_q$ polynomials of constant degree [29, 6, 7, 33]. Approximate counting algorithms with only a $1/\varepsilon^2$ dependence in the running time are not known for any of these problems except in the case of degree-two polynomials, where one can *exactly* count solutions over a finite field in polynomial time (see the Preliminaries). We would like to "lift" this nice algorithm to more expressive representations.

In the first part of this paper, we study approximate counting for an NP-hard problem in algebra:

Counting Solutions to Multivariate Quadratic Systems (#MQS)
Given: A system of degree-two polynomials over $\mathbb{F}_p[x_1, \ldots, x_n]$, for some prime p
Output: The number of solutions to the system.

The decision version of #MQS ("is there a solution?") is well-known to be NP-hard, and is of interest in several theoretical and practical areas (see [27] for references). With regards to approximating #MQS, the best deterministic algorithm in the literature is due to Viola [33], who gave a pseudorandom generator for fooling a degree-d polynomial with seed length at least $d \log n + d2^d \log(1/\varepsilon)$. Since the Fourier spectrum of the AND function has low ℓ_1-norm (see for example [9]), a pseudorandom generator for a single polynomial extends to a system of polynomials, yielding a deterministic approximation algorithm for the fraction of #MQS solutions with running time poly$(n) \cdot \varepsilon^{-8}$.

Approximately Solving #MQS

The first main result of this paper is that the $1/\varepsilon^2$ dependence of the random sampling algorithm for #MQS can be matched in a deterministic way.

▶ **Theorem 1.** *For every prime p, there is a deterministic algorithm running in poly$(n, s, \log p)/\varepsilon^2$ time for approximately counting the fraction of solutions to a system of s quadratic equations in n variables over $\mathbb{F}_p$.*

As a corollary, one can also efficiently and deterministically *find* solutions to any system of quadratic equations, provided there are many solutions:

▶ **Corollary 2.** *For every prime p, there is a deterministic algorithm running in poly$(n, s, \log p)/\varepsilon^2$ time which, given any system of s quadratic equations in n variables over $\mathbb{F}_p$ with at least εp^n solutions, outputs a solution.*

Recalling that more common counting problems such as #2-SAT and #NAE-3-SAT can easily be expressed as an instance of #MQS with no increase in the number of variables, Theorem 1 implies improved approximation algorithms (in terms of ε) for those problems as well: the best known approximate counting algorithm for general k-SAT is that of Trevisan [32] which has an $1/\varepsilon^{k2^k \ln(4)}$ dependence in the running time. (Note that Viola's

PRG is slightly faster in terms of ε.) For a non-binary example, the 3-coloring problem can be represented as a system of quadratic polynomials over $\mathbb{F}_3$ (for each edge (i, j) in your graph, include a polynomial $P(x_i, x_j)$ which is 0 if and only if $x_i \neq x_j$). In general, constraint satisfaction problems over a prime-order domain and two variables per constraint are handled by Theorem 1.

The key to Theorem 1 is a reduction from approximate counting for a system of degree-d polynomials to approximate counting for a *single* polynomial:

▶ **Theorem 3.** *For all primes p, integers $d > 0$, and $\varepsilon \in (0, 1)$, there exists a deterministic poly$(n, s, \log p)$-time reduction from approximately counting solutions to systems of degree-d equations over $\mathbb{F}_p$ to within an ε factor, to approximately counting solutions to* one *degree-d equation over $\mathbb{F}_p$ to within a $\Theta(\varepsilon^3/s^2)$ factor.*

Theorem 3 is appealing for several reasons.[1]

- First, it is somewhat surprising at first glance. For example, *detecting* feasibility of a system of quadratic polynomials (over any field) is NP-hard, but detecting the feasibility of only one such polynomial is trivial. Thus in some cases, our reduction efficiently reduces an NP-hard problem to an easy one — but only for the task of approximate counting.
- Second, the proof is extremely simple in hindsight. The central idea consists of applying the known constructions of *small-biased sets* in just the right way to solve the problem. We also show how such sets yield new schemes for approximating the intersection of a set family.
- Third, Theorem 3 is extremely generic, in comparison with its application (Theorem 1): it works for any polynomial system of *any* degree.

Theorem 1 follows from applying the reduction of Theorem 3 and an exact counting algorithm for #MQS instances with only one equation. We certainly do not obtain a pseudorandom generator in this way, but we do get a considerably different perspective on approximate counting in this domain. (We also do not use any algebraic geometry tools, which often appear in the literature on counting solutions to polynomial systems.)

SETH-Hardness of Exactly Counting Roots of One O(1)-Degree Polynomial

Complementing the above approximation algorithm, we use similar ideas to give a strong hardness reduction for exactly counting solutions (zeroes) of degree-d polynomials over a finite field of constant order, for constant $d > 2$. Finding a solution to such a polynomial is not hard: by a variant of the Schwartz-Zippel-DeMillo-Lipton lemma (see for example [36]), a not-identically-zero polynomial p of degree-d is nonzero on at least a $1/2^d$-fraction of points in $\{0, 1\}^n$, so it is easy to find a nonzero of p with randomness over any small field. (It is also easy to find a zero of p in $\{0, 1\}^n$ as well, by considering the polynomial $1 - p^{q-1}$ where q is the order of the field.)

Ehrenfeucht and Karpinski [17] showed that the solution-counting problem is already NP-hard for a single degree-3 polynomial over $\mathbb{F}_2$. However, all reductions from k-SAT to the counting problem (to our knowledge) increased the number of variables by a polynomial

[1] Note that over the real numbers, it is easy to reduce a system of polynomial equations $\{p_i(x_1, \ldots, x_n) = 0\}$ of degree d to a single equation of degree $2d$, by simply taking the sum of squares of the p_i. This is not useful for us, since (1) it does not work over a finite field and (2) in order for the approximation algorithm to work, we need a reduction that does not increase the degree.

factor; in the worst case, $\Omega(n)$ new variables are introduced. Here we give a much improved reduction in terms of the number of variables, sacrificing a bit in the degree:

▶ **Theorem 4.** *Let q be a prime power and $\varepsilon > 0$ be arbitrarily small. There is an $O(q^{\varepsilon n})$-time deterministic reduction from #k-SAT instances with n variables to the problem of counting roots to a $\mathbb{F}_q$-polynomial of degree $q(k/\varepsilon)^{O(k)}$ with n variables.*

In fact, the proof of Theorem 4 provides a subexponential-time reduction from the problem of exactly counting solutions to systems of $O(n)$ degree-k equations to that of exactly counting the zeros of one polynomial of degree $q(k/\varepsilon)^{O(k)}$, similarly to how Theorem 3 gives a polynomial-time reduction from approximate counting a degree-k system to approximate counting for a single degree-k polynomial. The high-level structure of the two reductions are similar as well: both reductions output a linear combination of the outputs of their oracle calls. However, the actual oracle calls themselves are quite different. Theorem 3 uses small-biased sets to construct a polynomial number of oracle calls, and obtain an approximate solution count; Theorem 4 uses a multiplication trick so that the number of oracle calls is only subexponential, while preserving the exact count. (The multiplication trick is the reason why the degree of the underlying polynomials increases to $q(k/\varepsilon)^{O(k)}$.)

From Theorem 4, it follows that the Strong Exponential Time Hypothesis (even its counting variant #SETH) predicts that for all $\varepsilon > 0$, there is a constant $d_\varepsilon > 2$ such that counting Boolean solutions to degree-d_ε polynomials (even over $\mathbb{F}_2$) cannot be done in $(2-\varepsilon)^n$ time.[2]

Under #ETH (the hypothesis that counting 3-SAT solutions requires $2^{\Omega(n)}$ time), it is known (for example) that counting the number of independent sets of size $n/3$ in an n-node graph requires $2^{\Omega(n)}$ time [24], #2-SAT requires $2^{\Omega(n)}$ time [15], the permanent of $n \times n$ Boolean matrices requires $2^{\Omega(n)}$ [15], and counting the number of perfect matchings in graphs with m edges requires $2^{\Omega(m)}$ time [14]. The reductions proving these conditional lower bounds generally introduce a minor (linear) increase in the relevant parameter n. Theorem 4 gives a tight lower bound for counting solutions to polynomials, under SETH.

Hardness for Counting Orthogonal Vectors over $\mathbb{F}_p$

Let p be any (small) prime. To demonstrate the range of the simple ideas in our reductions, we also use them to show that *exactly counting* pairs of $O(\log n)$-dimensional vectors with inner product zero *modulo* p (among a given set of n vectors) requires $n^{2-o(1)}$ time under SETH. This follows from the theorem:

▶ **Theorem 5.** *Let p be prime, and let $\ell \in [1, d]$ be an integer dividing d. There is an $\tilde{O}(n \cdot \ell 2^{d/\ell} \cdot p^\ell)$-time reduction from #OV with n vectors in d dimensions to p^ℓ instances of #OVp, each with n vectors in $\ell 2^{d/\ell} + 1$ dimensions.*

▶ **Corollary 6.** *Let $\varepsilon > 0$ be sufficiently small. There is an $\tilde{O}(n^{1+\varepsilon} \cdot p^{O(c/\varepsilon)})$-time reduction from #OV with n vectors in $c \log n$ dimensions to n^ε instances of #OVp, each with n vectors in $O(p^{c/\varepsilon} \log(n))$ dimensions.*

---

[2] The Strong Exponential Time Hypothesis [12] (SETH) states that for every $\varepsilon > 0$, there is a $k > 2$ such that k-SAT cannot be solved in $(2-\varepsilon)^n$ time; it is a vast strengthening of $\mathsf{P} \neq \mathsf{NP}$ and a mild strengthening of ETH [25] which states that there is an $\varepsilon > 0$ such that 3-SAT cannot be solved in $(1+\varepsilon)^n$ time. The "counting variant" #SETH states the same strong lower bound for the problem of *counting* the number of solutions to a k-SAT instance.

This reduction is in stark contrast with the fact that *detecting* a pair of vectors with inner product zero modulo p can be accomplished in nearly-linear time when the vectors have $n^{o(1)}$ dimension [35]. One can either view this result as evidence that the counting problem is hard, or as (yet) another angle towards refuting SETH.

2 Preliminaries

Exact counting for degree-two equations

Our approximate counting algorithm will use the fact that the exact counting problem for a single degree-two equation over a finite field $\mathbb{F}_q$ is polynomial-time solvable:

▶ **Theorem 7** (Woods [37], p.6). *For every prime power q, there is a deterministic $n^3 \cdot$ poly$(\log q)$-time algorithm for counting the number of solutions to a given degree-two polynomial over $\mathbb{F}_q$.*

Ehrenfeucht and Karpinski [17] covered the case of $\mathbb{F}_2$, and showed that exact counting for a degree-three polynomial is already NP-hard. Cai, Chen, Lipton, and Lu [10] gave an algorithm for the counting problem that works over $\mathbb{Z}_m$, for any fixed integer $m > 1$. Woods remarks that his algorithm essentially follows from placing the matrices defining the degree-two polynomial in Jordan canonical form, in which case the number of solutions can be more-or-less read off from the form obtained.

Small-bias sets

For our approximation algorithm, we need explicit constructions of small sets of vectors which closely approximate the uniform distribution of vectors, with respect to inner products over a finite field.

▶ **Definition 8.** A set $S \subseteq \mathbb{F}_q^n$ is *ε-biased* if for all $u \in \mathbb{F}_q^n$ and $r \in \mathbb{F}_q^n$,

$$\left| \Pr_{v \in S}[\langle u, v\rangle = r] - \Pr_{v \in \mathbb{F}_q^n}[\langle u, v\rangle = r] \right| \leq \varepsilon.$$

(Note that sometimes a more general definition is given, involving the characters of $\mathbb{F}_p^n$, but the above simple condition is implied by it. See [18].) The following simple consequences of being ε-biased will be important:

- $\Pr_{v\in S}[\langle \vec{0}, v\rangle = 0] = 1$, where $\vec{0}$ is the all-zeroes vector.
- For all $u \neq \vec{0}$ and $r \in \mathbb{F}_q$, $\Pr_{v\in S}[\langle u, v\rangle = r] \in (1/q - \varepsilon, 1/q + \varepsilon)$.

We also use deterministic explicit constructions of ε-biased sets over $\mathbb{F}_p$, for every prime p.

▶ **Theorem 9** ([4, 18, 5]). *For every prime p, and every $\varepsilon \in (0, 1/p^n)$, there is a (poly$(n, \log p)/\varepsilon^2$)-time constructible set of vectors $S \subseteq \mathbb{F}_p^n$ of cardinality $O(n^2/\varepsilon^2)$ that is ε-biased.*

It will be important that the ε-dependencies in the above theorem are only $1/\varepsilon^2$, but this can be achieved. For example, the constructions based on linear feedback shift registers of [4] (which are easily generalized to $\mathbb{F}_p$, see [18]) take all vectors $x, y \in \mathbb{F}_p^d$, where $d = \log_p(n/\varepsilon)$ and y corresponds to the coefficient vector of a monic irreducible $\mathbb{F}_p$-polynomial of degree d. The n-length vector $v_{x,y}$ of the ε-biased set S is generated by repeatedly computing inner products of y with vectors made up of previously computed inner products, up to n times.

To construct this set S, the main difficulty is constructing the y's, which we can do by enumerating all monic $\mathbb{F}_p$-polynomials of degree d, and throwing out those with non-trivial divisors. Rabin's test for irreducibility ([31]) would take $O(d^2 \cdot \mathrm{poly}(\log n, \log p))$ time. There are $O(n/\varepsilon)$ such polynomials to enumerate, and since $\varepsilon \geq 1/p^n$ we have $d \leq O(n)$, so this step takes $O(n^3/\varepsilon) \cdot \mathrm{poly}(\log n, \log p)$ time. The remaining list of monic irreducible polynomials (paired up with all possible vectors $x \in \mathbb{F}_p^d$) forms our ε-biased S, and each component of each n-length vector $v_{x,y}$ in S is just an inner product of two known d-length vectors. The running time of this construction is therefore $O(n^4/\varepsilon^2)$ (omitting $\mathrm{poly}(\log p)$ factors).

3 Approximating #MQS: Reduction and Algorithm

We begin with the proof of Theorem 3, which reduces the counting problem for a system of equations to the counting problem for a single equation.

Let $S = \{v_1, \ldots, v_m\} \subseteq \mathbb{F}_p^s$ be an ε-biased set of vectors. Let $\{p_1(y) = 0, \ldots, p_s(y) = 0\}$ be a system of degree-d equations over $\mathbb{F}_p$ in n variables, and let $A \subseteq \mathbb{F}_p^n$ be the set of solutions to the system. For each $i = 1, \ldots, m$, define the polynomial

$$P_i(y) = \sum_j v_i[j] \cdot p_j(y).$$

We observe two distinct properties of solutions and non-solutions to the original system:

(a) Every $y \in A$ is a solution of the equation $P_i(y) = 0$ and not of the equation $P_i(y) = 1$, for all $i = 1, \ldots, m$. This follows because for all $y \in A$, $p_1(y) = \cdots = p_s(y) = 0$.

(b) For every $y \notin A$, there are integers $N_0, N_1 \in [m(1/p - \varepsilon), m(1/p + \varepsilon)]$ such that y is a solution to N_0 of the m equations $P_i(y) = 0$ and is a solution to N_1 of the m equations $P_i(y) = 1$. To see this, note that for $y \notin A$, the vector $u = [p_1(y), \ldots, p_s(y)]$ is not all-zeroes. Thus for any $r \in \mathbb{F}_p$ we have $\Pr_{i \in [m]}[\langle u, v_i \rangle = r] \in (1/p + \varepsilon, 1/p - \varepsilon)$, because S is ε-biased.

Given the ability to count solutions to one degree-d equation, here is an algorithm for approximately counting solutions to a system of equations:

1. Construct the ε-biased set $S = \{v_1, \ldots, v_m\} \subset \mathbb{F}_p^s$.
2. Count the number of solutions to the equation $P_i(y) = 0$, for all $i = 1, \ldots, m$. Let Z be the sum of all these numbers.
3. Count the number of solutions to the equation $P_i(y) = 1$, for all $i = 1, \ldots, m$. Let O be the sum of all these numbers.
4. Output $(Z - O)/m$.

Now we analyze the algorithm. Let Z_i (respectively, O_i) be the number of solutions to the equation $P_i(y) = 0$ (respectively, $P_i(y) = 1$), for all $i, \ldots, m$. Our algorithm outputs the quantity:

$$\frac{1}{m}\left(\sum_i Z_i - \sum_i O_i\right). \tag{1}$$

By property (a), every $y \in A$ contributes 1 to the sum $\frac{1}{m} \cdot \sum_i Z_i$, and contributes 0 to the sum $\sum_i O_i$. By property (b), every $y \notin A$ contributes a value $z_y \in [1/p - \varepsilon, 1/p + \varepsilon]$ to the sum $\frac{1}{m} \cdot \sum_i Z_i$, and contributes a value $o_y \in [1/p - \varepsilon, 1/p + \varepsilon]$ to the sum $\frac{1}{m} \cdot \sum_i O_i$. We can therefore re-express (1) as:

$$\frac{1}{m}\left(\sum_i Z_i - \sum_i O_i\right) = |A| + \sum_{y \notin A}(z_y - o_y).$$

Given the bounds on z_y's and o_y's, we can easily upper-bound and lower-bound (1):

$$|A| + \sum_{y \notin A}(z_y - o_y) \leq |A| + \sum_{y \notin A}((1/p + \varepsilon) - (1/p - \varepsilon)) = |A| + |\overline{A}| \cdot 2\varepsilon$$

and

$$|A| + \sum_{y \notin A}(z_y - o_y) \geq |A| + \sum_{y \notin A}((1/p - \varepsilon) - (1/p + \varepsilon)) = |A| - |\overline{A}| \cdot 2\varepsilon.$$

It follows that the algorithm outputs a number that approximates the fraction of solutions to within $\pm 2\varepsilon$.

Moreover, observe that to obtain an approximate answer, we do not need an *exact* algorithm for counting solutions to one equation: if our algorithm for one equation always outputs approximations that are within $\varepsilon/(2m)$ of the exact count, then each of the $2m$ Z_i and O_i terms will be computed to within an $\varepsilon/(2m)$ factor, and the output will still be within $\pm 3\varepsilon$ of the exact fraction. This completes the proof of Theorem 3.

To obtain the final algorithm (Theorem 1) for approximately computing #MQS, we simply apply Theorem 7 to count the number of satisfying assignments to a single quadratic equation over $\mathbb{F}_p$ in $n^3 \cdot \text{poly}(\log p)$ time. Using this algorithm in the above reduction, we get an approximate counting algorithm running in time $\tilde{O}(s^2/\varepsilon^2 \cdot (n^3 + s) + t(s, 1/\varepsilon, p))$, where $t(s, 1/\varepsilon, p)$ is the time needed to construct an ε-biased set over $\mathbb{F}_p^s$. This completes the proof of Theorem 1.

3.1 A succinct approximate inclusion-exclusion

The reduction of Theorem 3 works by approximately representing the cardinality of the *intersection* of s equations by a linear combination of cardinalities on single $\mathbb{F}_p$-equations. Along the lines of the work of Linial and Nisan [26] on approximate inclusion-exclusion via low-degree polynomials over the reals, the ideas of Theorem 3 imply a variant of the inclusion-exclusion principle. However, unlike Linial and Nisan, our approximation of the cardinality of the intersection has only polynomially many terms.

To simplify the discussion, here we consider just the case of $\mathbb{F}_2$, and consider unions instead of intersections. Over $\mathbb{F}_2$, we will demonstrate how a small-bias set lets us "approximately" express the cardinality of a union of a set collection as a short linear combination of cardinalities of what one might call "oddtersections" of sub-collections of sets.

Let $(x \bmod 2) : \mathbb{Z} \to \{0, 1\}$ map integers to bits in the natural way. In Theorem 3, we are effectively using a representation of the AND function as a short linear combination of PARITY functions (see, for instance, Alon and Bruck [3]). Below is a representation of the OR function (which is analogous):

▶ **Lemma 10.** *For all $n \in \mathbb{N}$ and $\varepsilon \in (0, 1)$, there is a $(\text{poly}(n)/\varepsilon^2)$-time constructible collection of subsets $S_1, \ldots, S_m \subseteq [n]$, with $m \leq O(n^2/\varepsilon^2)$, such that for every $x \in \{0, 1\}^n$,*

$$\left|\left(\bigvee_{i=1}^{n} x_i\right) - \sum_{i=1}^{m} \frac{2}{m} \cdot \left(\sum_{j \in S_i} x_j \bmod 2\right)\right| \leq \varepsilon. \tag{2}$$

Proof. Let $\mathcal{S} = \{S_1, \ldots, S_m\} \subseteq [n]$ be a set family whose corresponding indicator vectors in $\{0,1\}^n$ form an $(\varepsilon/2)$-biased set. By Theorem 9, we can take $m \le O(n^2/\varepsilon^2)$. Observe:

- If $(x_1, \ldots, x_n) = \vec{0}$ then for all $i = 1, \ldots, m$, $(\sum_{j \in S_i} x_j \bmod 2) = 0$, so $\sum_{i=1}^m \frac{2}{m} \cdot \left(\sum_{j \in S_i} x_j \bmod 2\right) = 0$.
- If $(x_1, \ldots, x_n) \ne \vec{0}$ then by properties of $(\varepsilon/2)$-biased sets, the number of $i \in [m]$ such that $\sum_{j \in S_i} x_j \ne |S_i| \pmod 2$ is in the interval $[m/2 - \varepsilon m/2, m/2 + \varepsilon m/2]$. So in this case,

$$\sum_{i=1}^m \frac{2}{m} \cdot \left(\sum_{j \in S_i} x_j \bmod 2 \right) \in \left[\frac{2}{m} \cdot \left(\frac{m - \varepsilon m}{2} \right), \frac{2}{m} \cdot \left(\frac{m + \varepsilon m}{2} \right) \right]$$
$$= [1 - \varepsilon, 1 + \varepsilon] .$$

This completes the proof. ◀

Let $A_1, \ldots, A_k$ be any sets over a finite universe U, and define their *oddtersection* to be

$$\bigoplus_i A_i = \{x \in U \mid x \text{ appears in an odd number of the } A_i\text{'s}\}.$$

The upshot of Lemma 10 is that we can write:

$$\left| \bigcup_i A_i \right| \approx_\varepsilon \sum_{i=1}^m \frac{1}{m} \cdot \left| \bigoplus_{j \in S_i} A_j \right| , \tag{3}$$

where the $\approx_\varepsilon$ means that the two quantities are within $\varepsilon|U|$ of each other. (Note that $|\cup_i A_i|$ is the sum over all $y \in U$ of $\bigvee_{i=1}[y \in A_i]$, where $[y \in A_i]$ is 1 if $y \in A$, and 0 otherwise. Invoking (2) on each term in this sum, we obtain the right-hand side of (3) to within an additive $\pm\varepsilon|U|$ factor.) Thus we can approximately represent the cardinality of a union of sets in a sparse way, as "oddities" of various sub-collections. It seems likely that this observation has more applications. For example, equation (3) immediately implies that we can reduce approximate counting for k-DNF formulas (with additive error) to approximate counting for degree-k polynomials over $\mathbb{F}_2$ (with additive error), by letting A_i be the set of satisfying assignments to the ith clause of a DNF.

3.2 Producing a solution when there are many

Given Theorem 1, one can obtain a deterministic algorithm for producing a solution to a quadratic system given that it has many solutions, using a self-reducibility argument.[3]

Reminder of Corollary 2. *For every prime p, constant k, and fraction $\varepsilon \in [1/p^n, 1]$, there is a deterministic algorithm running in poly$(n, s, \log p)/\varepsilon^2$ time which, given any system of s quadratic equations in n variables over $\mathbb{F}_p$ with at least εp^n solutions, outputs a solution.*

Proof. Suppose we are given a system over the variables $x_1, \ldots, x_n$ with $S \ge \varepsilon \cdot p^n$ solutions, where $\varepsilon \ge 1/p^n$.

[3] This reduction is apparently folklore. See also Goldreich [19, Theorem 3.5] for a generic reduction from "search-to-decision" in this setting.

For each $a \in \mathbb{F}_p$, assign $x_1 := a$ in all equations of the system, and run the polynomial-time approximate counting algorithm of Theorem 1 with error parameter $\alpha := \varepsilon/(2n)$. Let $x_1 := a_1$ be the assignment that yields the largest count from the algorithm. (If the count returned is zero for all $a \in \mathbb{F}_p$, return *fail*.) Analogously, set the variables $x_2 := a_2, \ldots, x_{n-k} := a_{n-k}$ one at a time, for $k = 2\log_p(1/\varepsilon)$, always taking the assignment that yields the largest count. Finally, try all $p^k = p^{2\log_p(1/\varepsilon)} \leq 1/\varepsilon^2$ assignments on the remaining k variables, and return any solution found.

Given Theorem 1, it is clear that the algorithm runs in the desired time. Now we turn to correctness. The algorithm began with a guarantee of S solutions. At least one setting of the variable x_1 yields a system on $n-1$ variables with at least S/p solutions. So after setting x_1 to maximize the number of solutions returned by the algorithm, the number of solutions in the remaining $(n-1)$-variable system is at least $S_1 = S/p - \alpha \cdot p^{n-1}$. Similarly, after setting x_1 and x_2 appropriately, the number of solutions in the remaining system on $n-2$ variables is at least

$$S_2 = S_1/p - \alpha \cdot p^{n-2} = S/p^2 - \alpha \cdot p^{n-2} - \alpha \cdot p^{n-2} = S/p^2 - 2\alpha \cdot p^{n-2}.$$

After setting $x_1, \ldots, x_i$ for $i = 1, \ldots, n$, we are inductively guaranteed that the number of remaining solutions in the system is at least

$$S_i = S_{i-1}/p - \alpha \cdot p^{n-i} = S/p^i - \alpha \cdot i \cdot p^{n-i}.$$

For $i = n - 2\log_p(1/\varepsilon)$, the number of solutions remaining is at least

$$\varepsilon p^n/p^i - \alpha \cdot i \cdot p^{n-i} \geq \varepsilon \cdot p^{2\log_p(1/\varepsilon)} - \alpha \cdot np^{2\log_p(1/\varepsilon)} \geq (\varepsilon - \alpha n) \cdot 1/\varepsilon^2.$$

For $\alpha = \varepsilon/(2n)$, the number of solutions remaining after setting $x_1, \ldots, x_i$ is at least $1/\varepsilon^2 \cdot (\varepsilon/2) \geq 1/(2\varepsilon)$, i.e., the number is non-zero. Therefore the algorithm returns a solution, if there are at least εp^n solutions. ◀

4 From Counting k-SAT to Counting Roots to Polynomials of O(1)-Degree

Reminder of Theorem 4. *Let q be a prime power and $\varepsilon > 0$ be arbitrarily small. There is an $O(q^{\varepsilon n})$-time deterministic reduction from #k-SAT instances with n variables to the problem of counting roots to a $\mathbb{F}_q$-polynomial of degree $q(k/\varepsilon)^{O(k)}$ with n variables.*

Imagining q and k as fixed constants, and ε as a tiny parameter, we obtain a $2^{O(\varepsilon n)}$ time reduction from #k-SAT on n variables to counting roots of an $\mathbb{F}_q$-polynomial on n variables of degree poly$(1/\varepsilon)$.

Proof. Let $\varepsilon > 0$ be arbitrarily small. We are given a k-CNF formula F in n variables $x_1, \ldots, x_n$, and we want to reduce it to a single low-degree polynomial. We will in fact reduce the counting problem for F to a (sub-exponential) number of calls to counting roots of a single low-degree polynomial.

First, by the Sparsification Lemma [25, 11] (the counting version of which appears in [15]), we may assume without loss of generality that the k-CNF formula F has at most $m \leq (k/\varepsilon)^{O(k)}n$ clauses, with $2^{\varepsilon n}$-time overhead.

Second, we can express F as a system of m polynomial equations in the obvious way, where each equation contains at most k variables (and therefore each equation has degree at most k). For all $i = 1, \ldots, n$, add the degree-two equations $x_i \cdot (1 - x_i) = 0$ to the system (these equations simply force all solutions to be Boolean). Call the overall system of $m + n$ equations G, and note the number of solutions to G equals the number of SAT assignments to F.

Arbitrarily partition G into εn subsystems of equations $G_1, \ldots, G_{\varepsilon n}$, where each subsystem has at most $(k/\varepsilon)^{O(k)}$ equations. Our next move is to write each G_j as a *single* polynomial over the finite field $\mathbb{F}_q$. More precisely, given that G_j contains the $t = (k/\varepsilon)^{O(k)}$ equations

$$p_1(x_1, \ldots, x_n) = 0, \ldots, p_t(x_1, \ldots, x_n) = 0,$$

define the $(q-1)(k/\varepsilon)^{O(k)}$-degree polynomial

$$P_j(x_1, \ldots, x_n) := 1 - \prod_{i=1}^{t} (1 - p_i(x_1, \ldots, x_n)^{q-1}).$$

Note that for all $(a_1, \ldots, a_n) \in \mathbb{F}_q^n$, $P_j(a_1, \ldots, a_n) = 0$ if and only if $p_i(a_1, \ldots, a_n) = 0$ for each i with $1 \leq i \leq t$. Furthermore, since each p_i has at most k variables, there are at most kt variables in P_j. So by repeatedly applying the identity $x_i^q = x_i$ over $\mathbb{F}_q$, it takes no more than $q^{O(kt)} \leq q^{(k/\varepsilon)^{O(k)}}$ time to express the polynomial P_j as a sum of monomials, for all $j = 1, \ldots, \varepsilon n$.

Finally, we wish to exactly count the number of solutions to the system

$$P_1(x_1, \ldots, x_n) = 0, \ldots, P_{\varepsilon n}(x_1, \ldots, x_n) = 0 \tag{4}$$

where each P_j has $(k/\varepsilon)^{O(k)}$ variables and degree at most $q(k/\varepsilon)^{O(k)}$. Here, we reason similarly to the earlier approximate counting algorithm (namely, the reduction of Theorem 3), except instead of using small-biased sets of size polynomial in n, we simply use all $q^{\varepsilon n}$ possible linear combinations of the P_j's to exactly count.

For every $\beta \in \mathbb{F}_q^{\varepsilon n}$, suppose we count the number of zeroes to the polynomial

$$Q_\beta(x_1, \ldots, x_n) := \sum_{j=1}^{\varepsilon n} \beta_j \cdot P_j(x_1, \ldots, x_n)$$

and suppose we count the number of solutions to the polynomial $R_\beta := 1 - Q_\beta$. Note for all β, the degree of Q_β is at most $q(k/\varepsilon)^{O(k)}$. We want to show that a linear combination of these $O(q^{\varepsilon n})$ zero-counts will tell us the number of solutions to the original k-CNF F.

Suppose $(a_1, \ldots, a_n) \in \mathbb{F}_q^n$ is a solution to the system G. Then it is also a solution to the system (4). Hence $P_j(a_1, \ldots, a_n) = 0$ for all j, and therefore $Q_\beta(a_1, \ldots, a_n) = 0$ for all $\beta \in \mathbb{F}_q^{\varepsilon n}$. That is, $(a_1, \ldots, a_n)$ is a zero for all $q^{\varepsilon n}$ polynomials Q_β, and is never a zero for any R_β.

On the other hand, if $(a_1, \ldots, a_n) \in \mathbb{F}_q^n$ is not a solution to G, then $P_j(a_1, \ldots, a_n) \neq 0$ for some j. So we can think of each $Q_\beta(a_1, \ldots, a_n)$ as the inner product of the vector β with a fixed non-zero vector. Therefore in this case, $Q_\beta(a_1, \ldots, a_n) = 0$ for exactly $q^{\varepsilon n}/q$ polynomials Q_β, and $R_\beta(a_1, \ldots, a_n) = 0$ for exactly $q^{\varepsilon n}/q$ polynomials R_β.

Combining these observations, we conclude that

$$\frac{(\text{total number of zeros to all } Q_\beta) - (\text{total number of zeros to all } R_\beta)}{q^{\varepsilon n}}$$

equals the number of solutions to G. So we can solve #k-SAT by making $O(q^{\varepsilon n})$ calls to counting solutions to a single degree-$q(k/\varepsilon)^{O(k)}$ polynomial over $\mathbb{F}_q$. (Note that by tweaking ε slightly, we can write the number of calls as $O(2^{\varepsilon n})$.) ◀

We observe that the proof of Theorem 4 also provides a subexponential-time reduction from

exact counting for a system of $O(n)$ degree-$O(1)$ equations

to

exact counting for one degree-$O(1)$ equation.

(Referring back to the proof, even if each p_i depended on all n variables but had degree only k, each polynomial P_i would have n variables and degree at most $q(k/\varepsilon)^{O(k)}$, so it would take at most $n^{q(k/\varepsilon)^{O(k)}}$ time to expand each P_i into a sum of monomials.) To compare, Theorem 3 gave a polynomial-time reduction for the respective approximation versions (but from a degree-k system to a single degree-k polynomial).

4.1 A consequence for fine-grained counting complexity

The reduction method in the proof of Theorem 4 extends nicely to results on the fine-grained counting complexity of simple polynomial-time problems. Here we demonstrate this claim on the problem of counting the number of orthogonal pairs among a set of Boolean vectors:

> #Orthogonal Vectors (#OV)
> Given: vectors $v_1, \ldots, v_n, w_1, \ldots, w_n \in \{0,1\}^d$
> Output: The number of pairs (i,j) such that $\langle v_i, w_j \rangle = 0$.

Note that #OV is trivially solvable in $O(n^2 d)$ time, although faster algorithms are known for certain ranges of d [22, 13]. The detection problem OV (determining if there is at least one orthogonal pair) is widely studied. Finding a significantly faster algorithm for OV will already be challenging, as it is known that (for example) a $n^{1.9} \cdot 2^{o(d)}$ time algorithm for OV would contradict SETH [34]. A minor variant of OV studies the problem modulo a fixed prime p:

> Orthogonal Vectors Mod p (OVp)
> Given: vectors $v_1, \ldots, v_n, w_1, \ldots, w_n \in \mathbb{F}_p^d$
> Decide: Are there i, j such that $\langle v_i, w_j \rangle = 0 \bmod p$?

Williams and Yu [35] showed that OVp is apparently much easier than OV for constant p: it is solvable in $O(n \cdot d^{p-1})$ time.

One can similarly define #OVp, in which the task is to count the number of i, j such that $\langle v_i, w_j \rangle = 0 \bmod p$. Recently, Dell and Lapinskas [16] show how to use the algorithm for OVp to approximately compute #OVp efficiently. In particular, they show that for any $\varepsilon > 0$, given an #OVp instance with number of solutions N, one can output a value v such that $|v - N| \le \varepsilon N$ in $\tilde{O}(\varepsilon^{-4} n \cdot d^{p-1})$ time.

Interestingly, a minor modification of Theorem 4 shows that *exactly* computing #OVp is as hard as #OV itself:

Reminder of Theorem 5. *Let p be prime, and let $\ell \in [1, d]$ be an integer that divides d. There is an $\tilde{O}(n \cdot \ell 2^{d/\ell} \cdot p^\ell)$-time reduction from #OV with n vectors in d dimensions to p^ℓ instances of #OVp, each with n vectors in $\ell 2^{d/\ell} + 1$ dimensions.*

Proof. The idea of the reduction is analogous to Theorem 4, except we need to be slightly more abstract in our construction. We start with vectors $v_1, \ldots, v_n, w_1, \ldots, w_n \in \{0,1\}^d$, and we want to compute #OV on them. Partition the *components* of all vectors into ℓ parts, where each part has d/ℓ components. For each vector v_i, let $v_{i,1}, \ldots, v_{i,\ell} \in \{0,1\}^{d/\ell}$ be its decomposition into parts; define vectors $w_{i,j}$ similarly.

For each $j = 1, \ldots, \ell$, make a $2^{d/\ell}$-bit vector $a_{i,j}$ which has a 1 in the component corresponding to the d/ℓ-bit vector $v_{i,j}$, and 0s in all other components. Also for each $j = 1, \ldots, \ell$, make a $2^{d/\ell}$-bit vector $b_{i,j}$ which has a 1 for each d/ℓ-bit vector x such that $\langle x, w_{i,j} \rangle \neq 0$, and 0s everywhere else. (This is similar to "embedding 3" of Ahle, Pagh, Razenshteyn and Silvestri [1, Lemma 3].) Taking the vectors

$$a_i := (a_{i,1}, \ldots, a_{i,\ell}) \in \{0,1\}^{\ell 2^{d/\ell}}, b_i := (b_{i,1}, \ldots, b_{i,\ell}) \in \{0,1\}^{\ell 2^{d/\ell}},$$

over all $i = 1, \ldots, n$, we have $\langle v_i, w_j \rangle = 0$ if and only if $\langle a_i, b_i \rangle = 0$. Furthermore, notice that for all $j = 1, \ldots, \ell$, $\langle a_{i,j}, b_{i,j} \rangle \in \{0,1\}$, so for all primes p and for all j, we have $\langle a_{i,j}, b_{i,j} \rangle = 0$ if and only if $\langle a_{i,j}, b_{i,j} \rangle = 0 \bmod p$.[4]

We now build $2p^\ell$ instances of #OVp as follows. For every $\beta \in \mathbb{F}_p^\ell$, construct the $2n$ vectors

$$a_i^\beta = (\beta_1 a_{i,1}, \ldots, \beta_\ell a_{i,\ell}), b_i := (b_{i,1}, \ldots, b_{i,\ell}),$$

and let N_β be the number of pairs $(a_i^\beta, b_{i'})$ which are orthogonal modulo p, for all $i, i' = 1, \ldots, n$. Also construct

$$c_i^\beta = (1, \beta_1 a_{i,1}, \ldots, \beta_\ell a_{i,\ell}), d_i := (1, b_{i,1}, \ldots, b_{i,\ell}),$$

and let M_β be the number of pairs $(c_i^\beta, d_{i'})$ which are orthogonal modulo p. Our algorithm for #OV outputs the quantity

$$\sum_{\beta \in p^\ell} (N_\beta - M_\beta)/p^\ell.$$

It is easy to see that this reduction has the desired running time and number of oracle calls. We need to show that the reduction outputs the correct number of orthogonal pairs. For an orthogonal pair v_i, w_j in the original instance, we know that $\langle a_i, b_j \rangle = 0$, and therefore $\langle a_{i,j}, b_{i,j} \rangle = 0$ for all j. So for every vector β, $\langle a_i^\beta, b_j \rangle = 0 \bmod p$ as well. That is, every orthogonal pair v_i, w_j is counted p^ℓ times in the sum $\sum_\beta (N_\beta - M_\beta)$.

For an non-orthogonal pair v_i, w_j, we know that $\langle a_{i,j}, b_{i,j} \rangle \neq 0$ for some j. In particular, recalling that all $\langle a_{i,j}, b_{i,j} \rangle$ are either 0 or 1, we have that the ℓ-dimensional vector

$$ab_{i,j} = (\langle a_{i,1}, b_{i,1} \rangle, \ldots, \langle a_{i,\ell}, b_{i,\ell} \rangle)$$

is not the all-zero vector over $\mathbb{F}_p$. Observing that

$$\langle a_i^\beta, b_j \rangle = \langle \beta, ab_{i,j} \rangle \bmod p$$

and

$$\langle c_i^\beta, d_j \rangle = 1 + \langle \beta, ab_{i,j} \rangle \bmod p,$$

it follows that there are exactly $p^{\ell-1}$ choices of β for which $\langle a_i^\beta, b_j \rangle = 0 \bmod p$, and $p^{\ell-1}$ choices of β for which $\langle c_i^\beta, d_j \rangle = 0 \bmod p$. Therefore every non-orthogonal pair has a net contribution of zero to the sum $\sum_\beta (N_\beta - M_\beta)/p^\ell$. ◀

[4] Note [1] use the fact that $\langle a_i, b_i \rangle \in \{0, 1, \ldots, \ell\}$ to give a non-trivial inapproximability result for computing the maximum inner product between two vector sets.

Setting $\ell := \lceil \varepsilon \log_p(n) \rceil$ for tiny $\varepsilon > 0$, we obtain:

Reminder of Corollary 6. *Let $\varepsilon > 0$ be sufficiently small. There is an $\tilde{O}(n^{1+\varepsilon} \cdot p^{O(c/\varepsilon)})$-time reduction from $\#OV$ with n vectors in $c \log n$ dimensions to n^ε instances of $\#OVp$, each with n vectors in $O(p^{c/\varepsilon} \log(n))$ dimensions.*

Therefore an algorithm for counting orthogonal-mod-2 pairs in $n^{1.9} \cdot 2^{o(d)}$ time would yield a similar algorithm for counting orthogonal pairs, refuting SETH.

Acknowledgements. I am grateful to the Simons Institute at UC Berkeley for inviting me to the workshop on "Proving and Using Pseudorandomness" in Spring 2017. There, I got a chance to think more carefully about the ideas in the approximate counting algorithm. I also thank the reviewers of SOSA for some helpful comments, and Holger Dell for telling me about his work on approximate counting; the conversation prompted me to work out the details for Theorem 5.

References

1 Thomas Dybdahl Ahle, Rasmus Pagh, Ilya P. Razenshteyn, and Francesco Silvestri. On the complexity of inner product similarity join. In Tova Milo and Wang-Chiew Tan, editors, *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 151–164. ACM, 2016. doi:10.1145/2902251.2902285.

2 Miklós Ajtai and Avi Wigderson. Deterministic simulation of probabilistic constant depth circuits. *Advances in Computing Research*, 5:199–222, 1989.

3 Noga Alon and Jehoshua Bruck. Explicit constructions of depth-2 majority circuits for comparison and addition. *SIAM J. Discrete Math.*, 7(1):1–8, 1994. doi:10.1137/S0895480191218496.

4 Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple construction of almost k-wise independent random variables. *Random Struct. Algorithms*, 3(3):289–304, 1992. doi:10.1002/rsa.3240030308.

5 Yossi Azar, Rajeev Motwani, and Joseph Naor. Approximating probability distributions using small sample spaces. *Combinatorica*, 18(2):151–171, 1998. doi:10.1007/PL00009813.

6 Andrej Bogdanov and Emanuele Viola. Pseudorandom bits for polynomials. *SIAM J. Comput.*, 39(6):2464–2486, 2010. doi:10.1137/070712109.

7 Andrej Bogdanov and Emanuele Viola. Pseudorandom bits for polynomials. *SIAM J. Comput.*, 39(6):2464–2486, 2010. doi:10.1137/070712109.

8 Mark Braverman. Polylogarithmic independence fools AC^0 circuits. *J. ACM*, 57(5):28:1–28:10, 2010. doi:10.1145/1754399.1754401.

9 Jehoshua Bruck and Roman Smolensky. Polynomial threshold functions, ac^0 functions, and spectral norms. *SIAM J. Comput.*, 21(1):33–42, 1992. doi:10.1137/0221003.

10 Jin-yi Cai, Xi Chen, Richard J. Lipton, and Pinyan Lu. On tractable exponential sums. In Der-Tsai Lee, Danny Z. Chen, and Shi Ying, editors, *Frontiers in Algorithmics, 4th International Workshop, FAW 2010, Wuhan, China, August 11-13, 2010. Proceedings*, volume 6213 of *Lecture Notes in Computer Science*, pages 148–159. Springer, 2010. doi:10.1007/978-3-642-14553-7_16.

11 Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. A duality between clause width and clause density for SAT. In *21st Annual IEEE Conference on Computational Complexity (CCC 2006), 16-20 July 2006, Prague, Czech Republic*, pages 252–260. IEEE Computer Society, 2006. doi:10.1109/CCC.2006.6.

12 Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. The complexity of satisfiability of small depth circuits. In Jianer Chen and Fedor V. Fomin, editors, *Parameterized and Exact Computation, 4th International Workshop, IWPEC 2009, Copenhagen, Denmark, September 10-11, 2009, Revised Selected Papers*, volume 5917 of *Lecture Notes in Computer Science*, pages 75–85. Springer, 2009. doi:10.1007/978-3-642-11269-0_6.

13 Timothy M. Chan and Ryan Williams. Deterministic apsp, orthogonal vectors, and more: Quickly derandomizing razborov-smolensky. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1246–1255. SIAM, 2016. doi:10.1137/1.9781611974331.ch87.

14 Radu Curticapean. Parity separation: A scientifically proven method for permanent weight loss. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 47:1–47:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.47.

15 Holger Dell, Thore Husfeldt, Dániel Marx, Nina Taslaman, and Martin Wahlen. Exponential time complexity of the permanent and the tutte polynomial. *ACM Trans. Algorithms*, 10(4):21:1–21:32, 2014. doi:10.1145/2635812.

16 Holger Dell and John Lapinskas. Fine-grained reductions from approximate counting to decision. *CoRR*, abs/1707.04609, 2017. arXiv:1707.04609.

17 Andrej Ehrenfeucht and Marek Karpinski. The computational complexity of (xor, and)-counting problems. *Technical Report TR-90-031, International Computer Science Institute, Berkeley*, 1990. URL: http://www.icsi.berkeley.edu/pubs/techreports/tr-90-033.pdf.

18 Guy Even, Oded Goldreich, Michael Luby, Noam Nisan, and Boban Velickovic. Approximations of general independent distributions. In S. Rao Kosaraju, Mike Fellows, Avi Wigderson, and John A. Ellis, editors, *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pages 10–16. ACM, 1992. doi:10.1145/129712.129714.

19 Oded Goldreich. In a world of p=bpp. In Oded Goldreich, editor, *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation - In Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman*, volume 6650 of *Lecture Notes in Computer Science*, pages 191–232. Springer, 2011. doi:10.1007/978-3-642-22670-0_20.

20 Parikshit Gopalan, Adam R. Klivans, Raghu Meka, Daniel Stefankovic, Santosh Vempala, and Eric Vigoda. An FPTAS for #knapsack and related counting problems. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 817–826. IEEE Computer Society, 2011. doi:10.1109/FOCS.2011.32.

21 Parikshit Gopalan, Raghu Meka, and Omer Reingold. DNF sparsification and a faster deterministic counting algorithm. *Computational Complexity*, 22(2):275–310, 2013. doi:10.1007/s00037-013-0068-6.

22 Ben Gum and Richard J. Lipton. Cheaper by the dozen: Batched algorithms. In Vipin Kumar and Robert L. Grossman, editors, *Proceedings of the First SIAM International Conference on Data Mining, SDM 2001, Chicago, IL, USA, April 5-7, 2001*, pages 1–11. SIAM, 2001. doi:10.1137/1.9781611972719.23.

23 Edward A. Hirsch. A fast deterministic algorithm for formulas that have many satisfying assignments. *Logic Journal of the IGPL*, 6(1):59–71, 1998. doi:10.1093/jigpal/6.1.59.

24 Christian Hoffmann. Exponential time complexity of weighted counting of independent sets. In Venkatesh Raman and Saket Saurabh, editors, *Parameterized and Exact Computation - 5th International Symposium, IPEC 2010, Chennai, India, December 13-15, 2010. Proceedings*, volume 6478 of *Lecture Notes in Computer Science*, pages 180–191. Springer, 2010. `doi:10.1007/978-3-642-17493-3_18`.

25 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. `doi:10.1006/jcss.2001.1774`.

26 Nathan Linial and Noam Nisan. Approximate inclusion-exclusion. *Combinatorica*, 10(4):349–365, 1990. `doi:10.1007/BF02128670`.

27 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`.

28 Michael Luby and Boban Velickovic. On deterministic approximation of DNF. *Algorithmica*, 16(4/5):415–433, 1996. `doi:10.1007/BF01940873`.

29 Michael Luby, Boban Velickovic, and Avi Wigderson. Deterministic approximate counting of depth-2 circuits. In *Second Israel Symposium on Theory of Computing Systems, ISTCS 1993, Natanya, Israel, June 7-9, 1993, Proceedings*, pages 18–24. IEEE Computer Society, 1993. `doi:10.1109/ISTCS.1993.253488`.

30 Noam Nisan. Pseudorandom bits for constant depth circuits. *Combinatorica*, 11(1):63–70, 1991. `doi:10.1007/BF01375474`.

31 Michael O. Rabin. Probabilistic algorithms in finite fields. *SIAM J. Comput.*, 9(2):273–280, 1980. `doi:10.1137/0209024`.

32 Luca Trevisan. A note on approximate counting for k-dnf. In Klaus Jansen, Sanjeev Khanna, José D. P. Rolim, and Dana Ron, editors, *Approximation, Randomization, and Combinatorial Optimization, Algorithms and Techniques, 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2004, and 8th International Workshop on Randomization and Computation, RANDOM 2004, Cambridge, MA, USA, August 22-24, 2004, Proceedings*, volume 3122 of *Lecture Notes in Computer Science*, pages 417–426. Springer, 2004. `doi:10.1007/978-3-540-27821-4_37`.

33 Emanuele Viola. The sum of D small-bias generators fools polynomials of degree D. *Computational Complexity*, 18(2):209–217, 2009. `doi:10.1007/s00037-009-0273-5`.

34 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005. `doi:10.1016/j.tcs.2005.09.023`.

35 Ryan Williams and Huacheng Yu. Finding orthogonal vectors in discrete structures. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1867–1877. SIAM, 2014. `doi:10.1137/1.9781611973402.135`.

36 Virginia Vassilevska Williams, Joshua R. Wang, Richard Ryan Williams, and Huacheng Yu. Finding four-node subgraphs in triangle time. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1671–1680. SIAM, 2015. `doi:10.1137/1.9781611973730.111`.

37 Alan R. Woods. Unsatisfiable systems of equations, over a finite field. In *39th Annual Symposium on Foundations of Computer Science, FOCS '98, November 8-11, 1998, Palo Alto, California, USA*, pages 202–211. IEEE Computer Society, 1998. `doi:10.1109/SFCS.1998.743444`.