Approximating the Number of Solutions of a $GF[2]$ Polynomial

Marek Karpinski*
Department of Computer Science
University of Bonn
and
International Computer Science Institute
Berkeley, California

Michael Luby
International Computer Science Institute
Berkeley, California

Abstract

We develop a polynomial time Monte-Carlo algorithm for estimating the number of solutions to a multivariate polynomial over *GF[2]*. This gives the first efficient method for estimating the number of points on algebraic varieties over *GF[2]*, which has been recently proven to be $\#P$-complete even for cubic polynomials. There are a variety of applications of our result, which will be discussed in the full version of the paper.

1 Introduction

The problem of counting the number of points on algebraic varieties is a fundamental issue in algebra, geometry, and especially in algebraic geometry

*Supported in part by the DFG Grant KA 673/4-1, and by the SERC Grant GR-E 68297.

(which can be "defined" as the study of polynomial equations over fields). Among other things, this problem has direct applications in information and coding theory for computing weights of codes and channel-error probabilities (cf. [3], [7], [8]). For some special cases of counting the number of points on an algebraic variety, like counting the order of an elliptic curve over finite fields or the number of rational points on the Jacobian of a curve of genus two over finite fields, there are known polynomial time algorithms ([11], [1], [2]). However, in most cases there are no known efficient algorithms for counting exactly the number of points. Furthermore, for some cases the exact counting problem is known to be $\#P$-complete.

In lieu of the fact that exact counting is hard, the question arises: Is it possible to produce a reasonable estimate of the number of solutions to a general algebraic circuit over *GF(q)* in polynomial time? In this paper, we consider the special case of this problem when the input is a polynomial over *GF[2]*. Let

$$g(x_1, \ldots, x_n) = t_1(x_1, \ldots, x_n) \oplus t_2(x_1, \ldots, x_n) \oplus \cdots \oplus t_m(x_1, \ldots, x_n)$$

be a polynomial in variables $x_1, \ldots, x_n$ over *GF[2]*, where for each $i = 1, \ldots, m$, term $t_i(x_1, \ldots, x_n)$ is a product of a non-empty subset of the variables $x_1, \ldots, x_n$. Furthermore, for all $i \neq j$, $t_i \neq t_j$, i.e. there are no duplicate terms. Consider the equation

$$g(x_1, \ldots, x_n) = a,$$

where $a \in \{0, 1\}$. Let G be the subset of assignments to $x_1, \ldots, x_n$ that satisfy this equation. The *GF[2]* problem is to compute the number of assignments $|G|$ that satisfy this equation given a description of g and the value of a.[1]

The computational complexity of computing $|G|$ exactly over *GF[2]* has only recently been resolved. In [4] it is shown that the problem is $\#P$-complete. In fact, [4] shows that the problem is $\#P$-complete even when restricted to cubic polynomials over *GF[2]*. Degree three is the sharp bound for the intractability of exact counting for polynomials; an $O(n^3)$ time algorithm has been designed for exact counting for degree two polynomials over *GF[2]* [4].

[1] For brevity, hereafter the arguments $x_1, \ldots, x_n$ to the polynomial and to the terms of the polynomial are suppressed.

We develop a polynomial time Monte Carlo (ϵ, δ) approximation algorithm (cf.[5], [6]) for estimating $|G|$. We develop Monte Carlo algorithms A_0 and A_1 for estimating $|G|$ when $a = 0$ and $a = 1$, respectively. The analysis of the two algorithms, excluding the portion that relies on previous work, are very similar. We focus most of our attention on algorithm A_1 and, in a later section, we briefly describe algorithm A_0. Both algorithms have the following properties. Part of the input is a description of g, i.e. a list of the variables and for each term a list of the variables that appear in the term. The rest of the input is the allowable error ϵ in the output and the confidence level δ in the output. The output Y of the algorithm satisfies

$$\Pr[|G|(1-\epsilon) \leq Y \leq |G|(1+\epsilon)] \geq 1-\delta,$$

where this probability is taken over the random coin flips of the algorithm. The running time of A_0 is $O(nm^2 \ln(1/\delta)/\epsilon^2)$ and the running time of A_1 is $O(nm^3 \ln(1/\delta)/\epsilon^2)$.

Algorithm A_1 is based on the algorithm developed in [5] for estimating the number of truth assignments that satisfy a disjunctive normal form (*DNF*) formula. In the terminology used above, the difference between the *DNF* problem and the *GF[2]* problem is the following. For the *DNF* problem we want to estimate the fraction of assignments that satisfy at least one term, whereas for the *GF[2]* problem we want to estimate the fraction of assignments that satisfy an odd number of terms. Let $\bar{G}$ be the subset of assignments that satisfy at least one term of g, and as before let G be the subset of assignments that satisfy an odd number of terms of g. The additional component in the analysis of the *GF[2]* algorithm beyond the portion borrowed from the analysis of the *DNF* algorithm described in [5] is a combinatorial theorem which shows that $|\bar{G}|/|G| \leq m$.

Besides the basic algebraic interest in the *GF[2]* problem, it is also interesting from the viewpoint of circuit complexity. The polynomial g can be viewed as a depth two circuit over the basis $(\oplus, \wedge)$. Although we provide a randomized polynomial time approximation algorithm for the *GF[2]* problem, there are no known subexponential deterministic algorithms for approximating the *GF[2]* problem. On the other hand, for the related *DNF* problem when the basis is $(\vee, \wedge, \neg)$, in addition to the randomized polynomial time algorithm developed in [5], there is a deterministic approximation algorithm with running time $2^{\log^{O(1)} n}$ [9]. In fact, the results in [9] extend to the much more general case of a constant depth circuit over basis $(\vee, \wedge, \neg)$.

2 Algorithm A_1

In this section, we develop algorithm A_1. The outline of the algorithm is borrowed from the *DNF* approximation algorithm of [5]; we provide a self-contained description of the algorithm. We first review a simple and standard "dart throwing" algorithm that at a very high level provides the general outline for the *GF[2]* algorithm and highlights the design obstacles that must be overcome. We have a finite (but large) universe U of known size $|U|$, and our goal is to estimate the size of some set $G \subset U$ of unknown size. A trial of the algorithm for estimating $|G|$ consists of the following two steps:

(1) Randomly and uniformly choose $u \in U$ (i.e. throw a dart).

(2) See if $u \in G$ (i.e. see where the dart lands).

Let b be an easily computable upper bound on $|U|/|G|$. The algorithm performs $N = 4b\ln(2/\delta)/\epsilon^2$ independent trials, and the output Y is the fraction of these N trials where an element of G is chosen, multiplied by $|U|$. A standard analysis using an inequality due to Bernstein [10] shows that for $\epsilon < 1$,

$$\Pr[|G|(1-\epsilon) \leq Y \leq |G|(1+\epsilon)] \geq 1-\delta.$$

(See for example [6] for a proof.)

The key criteria in the design of the *GF[2]* algorithm are:

(a) The universe U should be defined in such a way that $|U|$ is easy to compute.

(b) Steps (1) and (2) above can be performed efficiently.

(c) $|G|$ is known a priori to be a significant fraction of $|U|$, i.e. b is polynomially bounded.

We now describe how to meet these criteria for algorithm A_1. For all $i = 1, \ldots, m$, let T_i be the set of all ordered pairs (i, s), where i is the index of term t_i and s is an assignment to $x_1, \ldots, x_n$ that that satisfies term t_i. Let $U = \cup_{i=1,\ldots,m} T_i$. Let $\bar{G} \subseteq U$ be the set of all pairs $(i, s) \in U$ such that there is no $j < i$ with $(j, s) \in U$. It is easy to verify that $|\bar{G}|$ is the number

of assignments that satisfy at least one term of g. Let $G \subseteq \bar{G}$ be the set of all pairs $(i, s) \in \bar{G}$ such that the number of terms that s satisfies is odd. It is easy to verify that $|G|$ is the number of assignments that satisfy an odd number of terms of g.

We now verify that (a), (b) and (c) are satisfied. By definition, $|U| = \sum_{i=1,\ldots,m} |T_i|$. For each $i = 1, \ldots, m$, $|T_i| = 2^{n-v_i}$, where v_i is the number of variables that appear in term t_i. Thus, $|U|$ can be easily computed. Furthermore, an element $(i, s) \in U$ can be uniformly and randomly chosen by the following two step process:

(i) Randomly choose $i \in \{1, \ldots, m\}$ with probability $|T_i|/|U|$.

(ii) Randomly choose $s \in T_i$ with probability $1/|T_i|$.

Also, $(i, s) \in G$ if and only if assignment s satisfies no term with index smaller than i and in total s satisfies an odd number of terms. Both of these conditions can be checked in $O(nm)$ time, and this is the most time consuming portion of a trial.

The final portion of the analysis is to show that there is an easily computable value b such that b is an upper bound on $|U|/|G|$ and such that b is polynomial in n and m. As shown in [5], it is not hard to verify that $|U|/|\bar{G}| \leq m$; this is simply because $|\bar{G}| \geq \max_{i=1,\ldots,m} |T_i|$ which implies that

$$|U|/|\bar{G}| \leq \sum_{i=1,\ldots,m} |T_i| / \max_{i=1,\ldots,m} |T_i| \leq m.$$

We prove in the next section that $|\bar{G}|/|G| \leq m$. It follows that $|U|/|G| \leq m^2$, and thus $N = 4m^2 \ln(2/\delta)/\epsilon^2$ trials suffice. Since the time per trial is $O(nm)$, the total running time of A_1 is $O(nm^3 \ln(2/\delta)/\epsilon^2)$.

3 The Main Theorem

In this section, we abuse the notation from the previous section slightly and let $\bar{G}$ be the subset of all assignments that satisfy at least one term of g, and let G be the set of assignments that satisfy an odd number of terms of g.

Theorem 1: $|\bar{G}|/|G| \leq m$.

Proof: The basic idea of the proof is to define a function $f : \bar{G} \rightarrow G$ in such a way that the mapping is at most m-to-1, i.e. for each $s \in G$, $|f^{-1}(s)| \leq m$. From this the theorem follows.

The mapping f is defined as follows. For each $\bar{s} \in \bar{G}$, choose any term t_i that is satisfied by $\bar{s}$ such that there is no term t_j which is satisfied and which contains all the variables in t_i. It is always possible to choose such a term because g does not contain two identical terms. Without loss of generality, let this be term t_1 and let $X = \{x_1, \ldots, x_k\}$ be the variables in t_1 (all of these variables are set to 1 in $\bar{s}$). For any $X' \subseteq X$, let $\bar{s}(X')$ be the assignment obtained from $\bar{s}$ by changing the values of all variables in $X - X'$ from 1 to 0.

Claim: There is at least one $X' \subseteq X$ such that $\bar{s}(X')$ satisfies an odd number of terms.

Proof of Claim: Let T be the set of terms satisfied by $\bar{s}$. If $|T|$ is odd, then let $X' = X$ and $\bar{s}(X') = \bar{s}$. If $|T|$ is even then we proceed as follows. Partition T into T' and T'', where T' is the set of terms that have at least one variable in common with X and T'' is the set of terms that have no variables in common with X. If $|T'|$ is odd (and hence $|T''|$ is odd) then we let $X' = \emptyset$, in which case $\bar{s}(X')$ satisfies none of the terms in T' and all the terms in T'', and thus an odd number of terms overall. If $|T'|$ is even (which means that $|T''|$ is also even), then we argue that there is an $X' \subseteq X$ such that $\bar{s}(X')$ satisfies an odd number of terms as follows.

For each $X' \subseteq X$, let $p(X')$ be the parity of the number of terms in T' that are satisfied by assignment $\bar{s}(X')$ and let $q(X')$ be the parity of the number of terms t_i in T' such that $t_i \cap X = X'$. By the way term t_1 is chosen, t_1 is the only term t_i that satisfies $t_i \cap X = X$, and thus $q(X) = 1$. We can view $p(\cdot)$ and $q(\cdot)$ as column vectors of length 2^k with entries from *GF[2]*, where the first entry corresponds to $X' = \emptyset$ and the last entry corresponds to $X' = X$. Then, it can be verified that there is a $2^k \times 2^k$ lower triangular matrix M over *GF[2]* with main diagonal 1 such that $M \cdot q(\cdot) = p(\cdot)$. In particular, row X' in M has a 1 in column X'' if and only if $X'' \subseteq X'$. (See Figure 1.)

Because M is invertible and because $q(\cdot) \not\equiv 0$, it follows that for at least one $X' \subseteq X$, $p(X') = 1$. For this X', $\bar{s}(X')$ satisfies an odd number of terms from T' and, as for $\bar{s}$, all the terms of T''. Thus, for this X', $\bar{s}(X')$ satisfies an odd number of terms overall. This complete the proof of the claim. □

	∅	{1}	{2}	{3}	{1,2}	{1,3}	{2,3}	{1,2,3}
∅	1	0	0	0	0	0	0	0
{1}	1	1	0	0	0	0	0	0
{2}	1	0	1	0	0	0	0	0
{3}	1	0	0	1	0	0	0	0
{1,2}	1	1	1	0	1	0	0	0
{1,3}	1	1	0	1	0	1	0	0
{2,3}	1	0	1	1	0	0	1	0
{1,2,3}	1	1	1	1	1	1	1	1

Figure 1: The matrix M for $k = 3$

We now complete the proof of the theorem. To define $f(\bar{s})$, we arbitrarily choose any X' such that $\bar{s}(X')$ satisfies an odd number of terms and let $f(\bar{s}) = \bar{s}(X')$. Finally, we argue that for each $s \in G$ there are at most m distinct assignments $\bar{s} \in \bar{G}$ such that $f(\bar{s}) = s$. This is because each such $\bar{s}$ is either equal to s, or is obtained by taking one of the terms not satisfied by s (there are at most $m - 1$, since s must satisfy an odd number and thus at least one term) and setting the values of all variables in this term to 1. □

It is not hard to see that the bound given in Theorem 1 is asymptotically optimal. To prove this, consider the following sequence of m-sparse polynomials. For m a power of two and for all $T \subseteq \{1, 2, \ldots, n\}$ such that $|T| = \log m$, let $g_T = \prod_{i \in T}(1 \oplus x_i) \prod_{j \notin T} x_j$. For each such T, g_T when expanded out has m terms. When viewed as a polynomial over *GF[2]*, $g_T = 1$ has a unique solution, whereas when viewed as a *DNF* formula, g_T has m satisfying assignments.

4 Algorithm A_0

In this section we briefly describe the algorithm A_0. In this case U is the set of all 2^n assignments to $x_1, \ldots, x_n$, and G is the set of all assignments that satisfy an even number of terms (possibly zero). A trial of the algorithm consists of choosing a random assignment s uniformly from U and seeing

if s satisfies an even number of terms. Each trial takes $O(nm)$ time. The output of the algorithm is the fraction of the assignments that satisfy an even number of terms, multiplied by $|U| = 2^n$. Theorem 2 below shows that $|U|/|G| \leq m$, and thus $N = 4m \ln(2/\delta)/\epsilon^2$ trials suffice. The overall running time of A_0 is $O(nm^2 \ln(2/\delta)/\epsilon^2)$.

Theorem 2: $|U|/|G| \leq m$

Proof: Similar to the proof of Theorem 1. □

5 Open Problems

The success of our method depends on a special property of the *GF[2]* field, and its connection to the *DNF* counting problem. An important open question is whether there is an (ϵ, δ) approximation algorithm for estimating the number of solutions of a multivariate polynomial over arbitrary finite fields *GF(q)* that runs in time polynomial in the problem input size, $1/\epsilon$, $\log(1/\delta)$ and $\log q$ (or even q). This problem is of paramount importance in algebra and algebraic geometry.

Another important question is whether it is possible to design a polynomial time deterministic approximation algorithm for the *GF[2]* (or more optimistically, for the *GF(q)* problem) that always produces an estimate with relative error at most ϵ. Such an algorithm would imply a deterministic polynomial time simulation of Monte Carlo depth two circuits over the $(\oplus, \wedge)$ basis.

6 Acknowledgements

We are indebted to Gunter Harder for the algebraic motivation of the problem, and to Dick Karp, Ronitt Rubinfeld and Volker Strassen for interesting discussions.

References

[1] Adelman, L.M., Huang, M.A., "Computing the Number of Rational Points on the Jacobian of a Curve", manuscript, 1987.

[2] Adelman, L.M., Huang, M.A., "Recognizing Primes in Random Polynomial Time", *Proc.* 19^{th} *ACM STOC (1987)*, pp. 462-469.

[3] Berlekamp, E.R., **Algebraic Coding Theory**, McGraw-Hill,1968.

[4] Ehrenfeucht, A., Karpinski, M., "The Computational Complexity of (XOR,AND)-Counting Problems", preprint, 1989.

[5] Karp, R., Luby, M., "Monte-Carlo Algorithms for Enumeration and Reliability Problems," 24^{th} *STOC*, November 7-9, 1983, pp. 54-63.

[6] Karp, R., Luby, M., Madras, N., "Monte-Carlo Approximation Algorithms for Enumeration Problems," *J. of Algorithms*, Vol. 10, No. 3, Sept. 1989, pp. 429-448.

[7] Kasami, T., Tokura, N., "On the Weight Structure of Reed-Muller Codes", *IEEE Trans. Inform. Theory IT-16(1970)*, pp. 752-759.

[8] MacWilliams, F.J., Sloan, N.J.A., **The Theory of Error Correcting Codes**, North-Holland, 1981.

[9] Nisan, N., Wigderson, A., "Hardness vs. Randomness", 29^{th} *FOCS (1988)*, pp. 2-11.

[10] Renyi, A., (1970), **Probability Theory**, North-Holland, Amsterdam.

[11] Schoof, R.J., "Elliptic Curves Over Finite Fields and the Computation of square Roots Mod p", *Math. Computation 44(1985)*, 483-494.