An Explicit Separation of Relativised Random Polynomial Time and Relativised Deterministic Polynomial Time

Richard Zippel

TR 89-965
February 1989

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

*Research on this paper was supported by DARPA grant N0014-88-K-0591, MSI grant U03-8300, NSF grant DMC-86-17355, and ONR grant N00014-86-K-0281.

An Explicit Separation of Relativised Random Polynomial Time and Relativised Deterministic Polynomial Time

RICHARD ZIPPEL
Department of Computer Science, Cornell University
Ithaca, New York 14853 *U. S. A.*

In this note we demonstrate that a certain class of naturally occuring problems involving an oracle are solvable in random polynomial time, but not in deterministic polynomial time. This class of problems is especially interesting because a very slight change in the parameters of the problem yields one that does have a polynomial solution.

1. Introduction

Recently there has been increased interest in the problem of interpolating a sparse polynomial P from its values. Efficient solutions to this problem can be used to simplify many multivariate polynomial calculations, including those as fundamental as computing the greatest common divisor of two polynomials. An essential component of determining P is the somewhat simpler of problem of showing that P is not the zero polynomial. In this paper we show that when only given certain information about P, no deterministic polynomial time algorithm can distinguish P from the zero polynomial. However, there do exist probabilistic algorithms that distinguish P from zero in polynomial time. Since proving P is or is not the zero polynomial is a special case of determining P, these results imply that P cannot be determined in deterministic polynomial time, and yet there exist probabilistic polynomial algorithms for this problem [8, 9].

To be more precise, let $P(X_1,\ldots,X_n) \in \mathbf{Z}[X_1,\ldots,X_n]$ be an n variable polynomial with rational integer coefficients and let $\mathcal{B}_P$ be a black box that represents P. $\mathcal{B}_P$ accepts an n-tuple $(a_1,\ldots,a_n) \in \mathbf{Z}^n$ and returns the value of $P(a_1,\ldots,a_n)$ in $\mathbf{Z}$. If $\mathcal{B}_P$ ever returns a non-zero value then P is not the zero polynomial. The *zero equivalence problem for* P is to provide an algorithm for generating n-tuples such that if P vanishes at each of the n-tuples, it is the zero polynomial.

To make this problem solvable bounds are needed that characterize the size of P. We are interested in algorithms whose time requirements are polynomial in terms of

these bounds. The most direct approach is to give bounds on the degrees of the X_i in P. We call this the *zero equivalence problem with degree bounds*. This problem was shown to be solvable in probabilistic polynomial time simultaneously by Schwartz [8] and Zippel [9] in 1979. Here, we show that this problem has no deterministic polynomial time solution.

Alternatively, we might be given a bound on the number of non-zero terms in the polynomial P instead of the degree bound. This problem is called the *zero equivalence problem with term bounds*. In 1987 ben Or and Tiwari [1] and Zippel [10] independently showed that this problem could be solved in deterministic polynomial time. Other recent work on this problem includes [2, 4, 5].

To minimize the number of subscripts in formulae we use a variant of Laurent Schwartz's notation. Let $\vec{X} = (X_1, X_2, \ldots, X_n)$ and $\vec{e} = (e_1, e_2, \ldots, e_n)$ be two vectors. Then we write the usual (inner) dot product as

$$\vec{e} \cdot \vec{X} = e_1 X_1 + e_2 X_2 + \cdots + e_n X_n.$$

We also extend this notation to exponentiation as follows

$$X^{\vec{e}} = (X^{e_1}, X^{e_2}, \ldots, X^{e_n}) \qquad \text{and} \qquad \vec{X}^{\vec{e}} = X_1^{e_1} X_2^{e_2} \cdots X_n^{e_n}.$$

Thus the multivariate polynomial

$$c_1 X_1^{e_{11}} X_2^{e_{12}} \cdots X_n^{e_{1n}} + c_2 X_1^{e_{21}} X_2^{e_{22}} \cdots X_n^{e_{2n}} + \cdots + c_t X_1^{e_{t1}} X_2^{e_{t2}} \cdots X_n^{e_{tn}}$$

will be written

$$c_1 \vec{X}^{\vec{e}_1} + c_2 \vec{X}^{\vec{e}_2} + \cdots + c_t \vec{X}^{\vec{e}_t}.$$

The vector accent is always given when this notation is used.

Since the black box accepts arguments that can be any rational integer, we need to consider the time $\mathcal{B}_P$ requires to compute its values. Otherwise, we might be able to use exceptionally large inputs to the black box to get more information than is justified. If the inputs to the black box are integers with absolute value less than 2^w, the time required for an evaluation by the black box is $O(w^r)$ for some r. If the black box implements a straightline algorithm and uses classical integer arithmetic, $r = 2$. If fast arithmetic algorithms are used $r = 1 + \epsilon$ for small ϵ. This estimate ignores the complexity of the computation performed by $\mathcal{B}_P$, but tries to take into account the growth in time required by larger inputs.

2. Main Results

The key observation in this paper is that it is easy to find non-zero polynomials that vanish at many points. This is shown in the following proposition.

Proposition 1. *Let D, T and n be integers such that $D^n \gg T$. Let $\mathcal{S} = \{\vec{a}_i\}$ be a set of $T - 1$ n-tuples. There exists a polynomial with rational integer coefficients with no more than T non-zero monomials and that vanishes at each point in $\mathcal{S}$.*

Proof: In fact, there are many polynomials that satisfy the requirements. Choose a set of T distinct n-tuples, $\vec{e}_1, \ldots, \vec{e}_T$. There exists a polynomial with these n-tuples as exponents that vanishes at every point in $\mathcal{S}$.

Let $P(\vec{X})$ be a polynomial whose monomials have $\vec{e}_1, \ldots, \vec{e}_T$ as exponent vectors:

$$P(\vec{X}) = c_1 \vec{X}^{\vec{e}_1} + c_2 \vec{X}^{\vec{e}_2} + \cdots + c_T \vec{X}^{\vec{e}_T}.$$

The coefficients (c_i) will be determined from $\mathcal{S}$. For P to vanish at $\vec{a}_i \in \mathcal{S}$ the c_j must satisfy the following linear equation:

$$c_1 \vec{a}_i^{\vec{e}_1} + c_2 \vec{a}_i^{\vec{e}_2} + \cdots + c_T \vec{a}_i^{\vec{e}_T} = 0.$$

Thus $P(\vec{X})$ vanishes at each element of $\mathcal{S}$ if the c_j satisfy the following system of linear equations.

$$\begin{aligned} c_1 \vec{a}_1^{\vec{e}_1} + c_2 \vec{a}_1^{\vec{e}_2} + \cdots + c_T \vec{a}_1^{\vec{e}_T} &= 0 \\ c_1 \vec{a}_2^{\vec{e}_1} + c_2 \vec{a}_2^{\vec{e}_2} + \cdots + c_T \vec{a}_2^{\vec{e}_T} &= 0 \\ &\vdots \\ c_1 \vec{a}_T^{\vec{e}_1} + c_2 \vec{a}_T^{\vec{e}_2} + \cdots + c_T \vec{a}_T^{\vec{e}_T} &= 0 \end{aligned}$$

Since these equations are homogenous and there are more variables than equations, the system possesses a non-trivial solution. □

Proposition 1 directly implies the non-existence of a deterministic algorithm for zero recognition.

Proposition 2. *Given a black box representing a polynomial $P(\vec{X})$ in n variables and of degree less than D in each variable, any deterministic algorithm the determines if P is the zero polynomial runs in time at least $O(D^n \log^r D)$.*

Proof: The time required by the algorithm is at least as large the number of trial evaluations used. Since no polynomial of degree less than D has more than D^n terms, D^n trials suffice. Thus the problem can be solved in exponential time. By proposition 1, if the algorithm uses less than D^n trials there will be polynomials that meet the degree bounds and that are indistinguishable from the zero polynomial. Thus at least D^n trial evaluations are required.

Since each of these trial points must be distinct, the components must be chosen from a set of at least D elements. Thus $\mathcal{B}_P$ will require $O(\log^r D)$ time for each trial, and any deterministic algorithm will need at least $O(D^n \log^r D)$ time. □

Though the zero equivalence problem, given degree bounds, is not solvable in deterministic polynomial time, notice that a polynomial that vanishes at each of $O(D^n)$ trial points, constructed as in the proof of proposition 2 would have $O(D^n)$ non-zero terms. It would be very interesting to know if there exists a polynomial that vanishes at those trial points and that has a more succint representation (straight-line program, for instance).

Probabilistic algorithms for the zero equivalence problem were initially introduced by Schwartz [8] and Zippel [9]. These algorithms require a random number generator.

Given an ϵ, this algorithm shows that P is zero with probability less than ϵ, averaged across random number generators with normal distributions. Such a probabilistic algorithm solves a problem in polynomial time if the time required is polynomial in the inputs and polynomial in ϵ^{-1}.

The probabilistic algorithms are based on the following proposition proved in [9, 10], where it is also shown to be the best possible result. The slightly weaker result of the last line of the proposition appears and is used in the work of Schwartz [8].

Proposition 3. *Let k be a field, $f \in k[X_1, \ldots, X_n]$ and the degree of f in each of X_i be bounded by d_i. Let $Z_n(B)$ be the number of zeroes of f, $\vec{x}$ such that x_i is chosen from a set with B elements, $B \gg d$. Then*

$$\begin{aligned} Z_n(B) &\leq B^n - (B - d_1)(B - d_2)\cdots(B - d_n) \\ &\approx O\left((d_1 + d_2 + \cdots + d_n)B^{n-1}\right). \end{aligned}$$

This proposition immediately gives a probabilistic algorithm for zero equivalence. We only indicate that a polynomial is non-zero when $\mathcal{B}_P$ returns a non-zero value, proving that P is non-zero. We want to know the probability that $\mathcal{B}_P$ returns zero even though P is not identically zero. Assume that P is not the zero polynomial.

Define the set $\mathcal{S}_B$ to be

$$\mathcal{S}_B = \{(x_1, \ldots, x_n) | 0 \leq x_i < B\}.$$

Let $\vec{x}$ be an element of $\mathcal{S}_B$ such that $\mathcal{B}_P$ returns zero. Then $\vec{x}$ is one of the at most $Z_n(B)$ zeroes in of P in $\mathcal{S}_B$. If we choose $\vec{x}$ randomly, the probability that we will get a zero of $P(\vec{X})$ is (by proposition 3)

$$\frac{Z_n(B)}{B^n} \leq \frac{(d_1 + d_2 + \cdots + d_n)}{B} = \frac{nD}{B},$$

where the $D \geq \max(d_i)$. The probability that k randomly chosen elements of $\mathcal{S}_B$ would both yield a value of zero (even though P is not the zero polynomial) is less than $(nD/B)^k$.

To verify that a polynomial $P(X_1, \ldots, X_n)$, whose degree in each variable is less than D, is zero with probability less than ϵ we need to perform k evaluations with random elements of $\mathcal{S}_B$ where

$$\epsilon < \left(\frac{nD}{B}\right)^k. \tag{1}$$

A single evaluation will suffice if B is chosen such that $B \geq nD/\epsilon$. The cost of the single evaluation is the cost of the black box evaluation. Recall that the cost of a black box evaluation with n-tuple components of size w is $O(w^r)$, for some r. Since the size of the components of the random n-tuple is $\log nD/\epsilon$, the cost of a single evaluation is $O(\log^r nD/\epsilon)$. If k different evaluations are used, then we need to choose B such that (1) is satisfied. So,

$$\log B < \log nD + \frac{1}{k}\log\frac{1}{\epsilon}.$$

The cost of a single evaluation when k are intended is

$$O\left(\log^r nD\left(\frac{1}{\epsilon}\right)^{1/k}\right),$$

and the total time required for k evaluations using the probabilistic approach will be

$$C_{\text{prob}} = O\left(k\log^r nD\left(\frac{1}{\epsilon}\right)^{1/k}\right).$$

This quantity is minimized when

$$k = \frac{(r-1)\log\frac{1}{\epsilon}}{\log nD}.$$

Since $1/\epsilon \gg nD$, using multiple evaluations with relatively small evaluation points is better than a single evaluation at a large evaluation point.

Replacing k by this quantity in C_{prop} we have

$$\begin{aligned} C_{\text{prob}} &= O\left(\frac{(r-1)\log\frac{1}{\epsilon}}{\log nD}\log^r nD\left(\frac{1}{\epsilon}\right)^{\frac{\log nD}{(r-1)\log\frac{1}{\epsilon}}}\right) \\ &= O\left(\frac{\log\frac{1}{\epsilon}}{\log nD}\log^r(nD)^{1+\frac{1}{r-1}}\right) \\ &= O(\log\frac{1}{\epsilon}\times\log^{r-1} nD). \end{aligned}$$

Notice that the cost of the probabilistic algorithm is linear in the logarithm of of the error. This, I feel, is a good characterization of what it means for a probabilistic algorithm to be polynomial time.

There is a slight variant of the zero equivalence problem that *can* be solved in deterministic polynomial time. Instead of being given degree bounds for the polynomial, we are given a bound on the number of non-zero monomials in the polynomial. We call this problem the *zero equivalence problem with term bounds.*

The following proposition and the algorithm based on it are due to Gregoriev and Karpinski [3].

Proposition 4. (Grigoriev and Karpinski) *Let* $P(\vec{X}) \in \mathbf{Z}[X_1,\ldots,X_n]$ *and assume that* P *has no more than* T *monomials. Then there exists a set of* T n*-tuples such that* P *either is different from zero at one of them or* P *is identically zero.*

Proof: Write P as

$$P(\vec{X}) = a_1\vec{X}^{\vec{e}_1} + \cdots + a_T\vec{X}^{\vec{e}_T},$$

where some of the a_i may be zero. If we evaluate P at $(2,3,5,\ldots,p_n)$ (where p_n is the n-th prime) we will have

$$\begin{aligned} P(2,3,\ldots,p_n) &= a_1(2,3,\ldots,p_n)^{\vec{e}_1} + a_2(2,3,\ldots,p_n)^{\vec{e}_2} + \cdots + a_T(2,3,\ldots,p_n)^{\vec{e}_T} \\ &= a_1 m_1 + a_2 m_2 + \cdots + a_T m_T, \end{aligned}$$

where the m_i are dstinct because of unique factorization. Using $\vec{t}_k = (2^k, 3^k, \ldots, p_n^k)$ as our evaluation points,

$$P(\vec{t}_k) = a_1 m_1^k + a_2 m_2^k + \cdots + a_T m_T^k.$$

If P is zero at each of $\vec{t}_0, \ldots, \vec{t}_{T-1}$, then the coefficients of P, a_i, will satisfy the following system of linear equations:

$$\begin{aligned} a_1 + a_2 + \cdots a_T &= 0 \\ a_1 m_1 + a_2 m_2 + \cdots + a_T m_T &= 0 \\ a_1 m_1^2 + a_2 m_2^2 + \cdots + a_T m_T^2 &= 0 \\ \vdots \qquad\qquad & \vdots \\ a_1 m_1^{T-1} + a_2 m_2^{T-1} + \cdots + a_T m_T^{T-1} &= 0 \end{aligned}$$

This is a Vandermonde system and is nonsingular since each of the m_i are different. Thus each either all of the a_i are zero or P is different from zero and will not vanish at one of $\vec{t}_i$. □

A deterministic solution of the zero equivalence problem with term bounds follows from proposition 4 easily. We pass the trial points $\vec{t}_0, \ldots, \vec{t}_{T-1}$ to $\mathcal{B}_P$. If $\mathcal{B}_P$ returns an answer different from zero for any of the $\vec{t}_i$ then P is not the zero polynomial. If $\mathcal{B}_P$ returns zero for all of the trial points then P is the zero polynomial. This algorithm requires T evaluations using $\mathcal{B}_P$, the last of which will involve numbers as large p_n^{T-1}. The n-th prime is approximately $n \log n$. So the k-th evaluation by $\mathcal{B}_P$ will require $O((k \log(n \log n))^r)$ time, which we approximate by $O((k \log n)^r)$. Thus all T evaluations require

$$\sum_{k=0}^{T-1} O((k \log n)^r) = O(T^{r+1} r^{-1} \log^r n) = O(T^{r+1} \log^r n)$$

time.

3. Discussion

The results of the previous section are summarized in the following table. The columns correspond to probabilistic and deterministic algorithms respectively. The rows correspond to whether degree bounds or term bounds are given. In the deterministic, bounded degree case, we have given a lower bound on the time required, while for the other two entries we have demonstrated algorithms that achieve the indicated

performance. Also recall that r is a constant corresponding to the type of arithmetic being used by $\mathcal{B}_P$. For classical arithmetic $r = 2$; for fast arithmetic r is slightly greater than 1.

	Probabilistic	Deterministic
degree bounds	$\log \frac{1}{\epsilon} \times \log^{r-1} nD$	$D^n \log^r D$
term bounds		$T^{r+1} \log^r n$

As remarked earlier, the zero equivalence problem with degree bounds cannot be solved in deterministic polynomial time, yet it can be solved in probabilistic polynomial time. We find it very curious that with a slight change to the way information is provided, the problem can be solved in determinstic polynomial time. The zero equivalence problem seems lies on the boundary between those problems that can and cannot be solved in deterministic polynomial time. Though earlier work has shown that random and deterministic polynomial time can be separated via an oracle [6], we find this example interesting because it arises in a common practical problem.

By representing the polynomial as a black box, we have swept the issues of the size of the computation required to compute $P(\vec{x})$ under the rug. If we could look inside $\mathcal{B}_P$ and examine the "program" used to compute $P(\vec{x})$ we might be able to show that $\mathcal{B}_P$ represents the zero polynomial without any bounds on P. For example, it seems likely that one can deterministically prove that a straightline program for a polynomial (in the sense of Kaltofen [7]) can be deterministically shown to represent the zero polynomial in time polynomial in the size of the straightline program.

4. Acknowledgements

This work was motivated by a comment by Erich Kaltofen, and benefited from useful discussions with Shafi Goldwasser.

This report describes research done jointly at the Artificial Intelligence Laboratory and the Laboratory for Computer Science of the Massachusetts Institute of Technology. Support for the Laboratory's artificial intelligence research is provided by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014–86–K–0180. Support for work in the Laboratory for Computer Science is provided by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014–83–K–0125.

Preparation of this report was supported by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014–86–K–0591, the Mathematical Sciences Institute under contract U03–8300, the National Science Foundation through contract DMC–86–17355 and the Office of Naval Research through contract N00014–86–K-0281.

References

1. Ben Or, M. and Tiwari, P., "A Deterministic Algorithm for Sparse Multivariate Polynomial Interpolation," *STOC '88,* (1988), 301–309.

2. M. Clasen, A. Dress, J. Grabmeier and M. Karpinski, *On Zero-Testing and Interpolation of k-Sparse Multivariate Polynomials over Finite Fields,* Research Report No. 8522–CS, University of Bonn, May, 1988 .
3. D. Yu. Grigoriev and M. Karpinski, "The Matching Problem for Bipartite Graphs with Polynomial Bounded Permanents is in NC," *Proceedings of 28th IEEE Symposium on the Foundations of Computer Science,* (1987), 166–172.
4. D. Yu. Grigoriev, M. Karpinski and M. F. Singer, *Fast Parallel Algorithms for Sparse Multivariate Polynomial Interpolation over Finite Fields,* Research Report No. 8523–CS, University of Bonn, May, 1988 .
5. E. Kaltofen and L. Yagati, *Improved Sparse Multivariate Polynomial Interpolation Algorithms,* 88–17, Depart. of Computer Science, Rensselaer Polytechnic Institute, Troy, New York, (1988) .
6. H. Heller, "On Relativized Exponential and Probabilistic Complexity Classes," *Information and Control,* **2**, (1986), 231–243.
7. Kaltofen, E., "Computing with Polynomials given by Straight-Line Programs I; Greatest common divisiors," *Proceedings, 17th ACM Symposium on Theory of Computation,* (1985), 131–142.
8. Schwartz, J. T., "Probabilistic Algorithms for Verification of Polynomial Identities," *Journal of the ACM,* **27**, (1980), 701–717.
9. Zippel, R. E., "Probabilistic Algorithms for Sparse Polynomials," *Lecture notes in Computer Science 72: Symbolic and Algebraic Computation,* Springer-Verlag, New York, (1979), 216-226.
10. Zippel, R. E., "Interpolating Polynomials from their Values," *Journal of Symbolic Computation,* to appear.