J. Symbolic Computation (1990) 9, 375–403

Interpolating Polynomials from their Values

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

(*Received* 4 *September* 1989)

A fundamental technique used by many algorithms in computer algebra is interpolating polynomials from their values. This paper discusses two algorithms for solving this problem for sparse multivariate polynomials, an updated version of a probabilistic one and a new deterministic technique that uses some ideas due to Ben-Or and Tiwari (1988). In addition algorithms are presented for quickly finding points that are not zeroes of sparse multivariate polynomials—the zero avoidance problem.

1. Introduction

Mathematical calculations involving polynomials or other symbolic quantities suffer from a problem not found in numerical calculations: *intermediate expression swell*. That is, when performed in a straightforward fashion, the intermediate expressions of a calculation are much larger than the final answer. Fundamentally, this difference is due to the fact that the amount of space required to represent the product of two floating point numbers is about as much as for each of the original multiplicands. However, the space required for the product of two multivariate polynomials can be much larger than that required for the multiplicands. In fact, even the sum of two multivariate polynomials can be twice as large the summands. This effect is more pronounced with polynomials with many variables.

Two fundamental approaches to this problem have been suggested. Each generates one or more simplified computations where some of the symbolic variables are replaced by numerical values. These simplified problems do not suffer as much from intermediate expression swell and may be solved more easily than the original problem. The two techniques differ in how they determine the solution of the original problem from the solutions of the simplified ones.

The first approach, which we call the *modular technique*, solves a large number of simplified problems but uses carefully chosen values for the symbolic variables. These solutions are then interpolated to recover the variables eliminated in the simplified problems, producing the final answer. In many practical algorithms the

© 1990 Academic Press Limited

resulting intermediate expressions do not involve any symbolic variables and there is essentially no intermediate expression swell. This interpolation technique was first introduced in the modular GCD algorithm of Brown (1971).

The second approach, which we call *Newton's technique*, uses the solution to a single simplified problem as a the initial value for a p-adic solution derived by Newton's iteration. (Conversion of a p-adic solution to a solution in the original ring is rarely difficult.) This is the basic idea behind the polynomial factoring algorithm of Wang and Rothschild (1975), the EZGCD algorithm of Moses and Yun (1973) and its successors Wang (1978) and most of the polynomial factoring algorithms now in use. Both the modular technique and Newton's technique suffer when the answer is sparse (has relatively few non-zero coefficients). In this case a great deal of effort is expended computing coefficients that are zero.

Versions of both the modular technique and Newton's technique whose time complexity is random polynomial were first given in Zippel (1979, 1980). Applications of these techniques to polynomial factoring and their analysis and extension, have been presented in a number of papers by von zur Gathen and Kaltofen: von zur Gathen (1983, 1985), Kaltofen (1985a, 1985b, 1987) and von zur Gathen and Kaltofen (1985). The probabilistic nature of these algorithms stems from an assumption about certain polynomials that arise in the calculation. It is known that the values of these polynomials at certain points are zero. This could happen either if the polynomials were identically zero or if the points chosen happened to be zeros of the polynomials. The key assumption of these algorithms is that the polynomials are identically zero. These algorithms can be made deterministic by choosing points that cannot all be zeroes of these polynomials. We call this the *zero avoidance problem.*

Problem. (Zero Avoidance Problem) *Given some set of parameters for a polynomial (number of variables, degree, number of non-zero terms, size of coefficients, etc.) choose a set of points S such that no polynomial with those parameters vanishes at all of the points of S.*

The original sparse polynomial algorithms used only the number of variables (n) and degree (d) parameters in choosing the set S. It is easy to show that S must contain at least $(d+1)^n$ points (see proposition 1 in section 2). To prove that a polynomial is zero using this set of points would require time exponential in the number of variables. Thus fast algorithms that use only these parameters are probabilistic. The deterministic algorithms given here also make use of the number of non-zero terms (T) in choosing S. It is this additional bit of information about the polynomial that keeps the size of S small.

Many of the ideas used to solve the zero avoidance problem can be used to clarify and simplify certain steps in the modular technique. The particular piece that we discuss in this paper we call the *interpolation problem.* Rather than choosing points to prove that a polynomial is not identically zero, we go further and actually determine the polynomial itself.

Problem. (Interpolation Problem) *Given a set of parameters for a polynomial (number of variables, degree, number of non-zero terms, size of coefficients, etc.) choose a set of points S with the following property. For any polynomial P with those parameters, P can be determined from S and $P(S)$ quickly, i.e. either polynomial time or probabilistic polynomial time.*

In this paper we present three solutions to the zero avoidance problem, and two solutions to the interpolation problem. Each is summarized in the following two tables.

Zero Avoidance Problem			
	Schwartz	Zippel	Ben-Or Tiwari
Algorithm type	Probabilistic	Deterministic	Deterministic
Number of evaluations	1	nT^2	$T+1$
Chance of error	ϵ	0	0
Size of evaluations in bits	$\log \frac{nd}{\epsilon}$	$nT \log T$	$T \log n$

The column labeled "Schwartz" corresponds to the probabilistic algorithm presented by J. Schwartz (1980) and which is intrinsic to Zippel (1979). Since it does not take into consideration the number of non-zero terms in the P, the parameter T does not appear. In the third column, labeled "Ben-Or Tiwari," we give the recent results of Ben-Or and Tiwari (1988). The second column, labeled "Zippel," is a new algorithm presented here in section 5. Though its performance is inferior to that of Ben-Or and Tiwari it makes use of some new techniques that may be of use in other problems. In particular, it yields a deterministic solution of a variant of the zero avoidance problem for polynomials over finite fields.

For the interpolation problem a new parameter arises, t the true number of non-zeroes terms in P. This can be much smaller than the *a priori* bound on the number of non-zero terms T.

The first column of this table characterizes the author's original probabilistic algorithm updated to include an idea of Ben-Or and Tiwari. The third column corresponds to the deterministic algorithm due to Ben-Or and Tiwari (1988). It is unique in that it does not require a *priori* bounds on the degrees of the variables that appear in the result. Notice that the probabilistic algorithm is significantly better than the deterministic one when the bound on the number of terms is not sharp ($T \gg t$). The second "Zippel" algorithm is a new deterministic variant

Interpolation Problem			
	Zippel	Zippel	Ben-Or Tiwari
Algorithm type	Probabilistic	Deterministic	Deterministic
Degree bounds	Yes	Yes	No
Number of operations	ndt^2	ndt^2T	$T^2(\log^2 T + \log nd)$
Number of evaluations	ndt	$ndtT$	$2T$
Size of evaluations in bits	$\log \frac{nd^2T^2}{\epsilon}$	$T \log n$	$T \log n$

of the probabilistic algorithm whose dependence on T is not quite so strong as Ben-Or and Tiwari's algorithm. Thus it also performs especially well when T is not a sharp bound. This algorithm is presented in section 6.

Kaltofen and Yagati (1988) have suggested an improved technique for solving the systems of linear equations that arise in the two interpolation algorithms discussed in this paper. Their ideas improve the algorithms discussed in the paper to give the performance figures given above. In this table $M(t)$ denotes the complexity of multiplying two univariate polynomials of degree t. This variant of the deterministic algorithm is competitive with Ben-Or and Tiwari's algorithm.

Interpolation Problem		
	Kaltofen-Yagati	Kaltofen-Yagati
Algorithm type	Probabilistic	Deterministic
Degree bounds	Yes	Yes
Number of operations	$ndM(t) \log t$	$ndTM(t) \log t$
Number of evaluations	ndt	$ndtT$
Size of evaluations in bits	$\log \frac{nd^2T^2}{\epsilon}$	$T \log \log n$

In the conclusions we give some comments on how these algorithm impact some of the original calculations, such as greatest common divisor and factorization problems.

2. Generalities

We let $\mathbf{Z}$ denote the rational integers and $\mathbf{Z}/(m)$ the integers modulo m. $\mathbf{F}_q$ denotes the finite field with q elements and $\mathbf{F}_q^*$ its multiplicative subgroup.

Throughout this paper we assume polynomials are represented as a list of monomials (pairs of exponent vectors and coefficients) and that monomials with zero coefficients are omitted. The number of variables in a polynomial is denoted by n. Thus the exponent vectors are n-tuples. The maximum degree of any variable in the polynomial is denoted by d. The number of non-zero monomials of the polynomial P is usually denoted by t or terms(P), for additional preciseness. For a *dense polynomial*, one where each monomial has a non-zero coefficient, terms$(P) = (d+1)^n$. We generally use capital letters to denote *a priori* bounds, and lower case letters for the actual value. Thus T is used to designate a bound on the number of terms in P, while t denotes the actual number of non-zero terms present in P.

To minimize the number of subscripts in formulae we use a variant of the notation introduced by Laurent Schwartz. 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

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

would be written

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

We always use the vector accent when using this notation.

When evaluating algorithms involving polynomials, we need to measure the size of a polynomial. In this paper we have chosen to use the number of non-zero terms. Thus an upper bound on the size of a polynomial of n variables, each of degree d, is $(d+1)^n$. The number of non-zero terms, however, is often much smaller. Notice that when establishing that a polynomial P of size $O(T)$ is identically zero, we already know that P cannot have more than $O(T)$ non-zero coefficients, though we know little about the exponents.

An alternative measure of the size of a polynomial P is the size of a straight line program to compute P. This measure was advanced by Kaltofen (1987). The class of straight line programs of size $O(T)$ contains almost all polynomials with $O(T)$ non-zero terms and many more. It would be interesting to know if it is possible to extend the results presented here to this wider class of polynomials.

To prove that a polynomial is zero by considering its value at a number of points requires some bound on the information content of the polynomial. We begin with a proposition that establishes a lower bound for our results.

Proposition 1. *Let $S = \{\vec{a}_i\}$ be a set of $T-1$ n-tuples. There exists a polynomial with rational integer coefficients, not identically zero, that contains no more than T non-zero monomials and that vanishes at every point in S.*

Proof: Choose a polynomial with T monomials:

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

whose coefficients (c_i) will be determined later from S, $\vec{e}_i \neq \vec{e}_j$ and chosen arbitrarily. For P to vanish at $\vec{a}_i$, an element of S, the c_j must satisfy the following linear equation:

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

Since these equations are homogeneous and there are more undetermined variables than linear constraints, there is a non-trivial solution to this system of equations. □

Assume we wish to prove that a polynomial is zero using only its value at points that we are free to specify. Proposition 1 demonstrates that showing the polynomial is zero at T points only shows that that the polynomial is either identically zero or has more than T non-zero terms. Thus if all that is known about a polynomial is the number of variables, n and degree bounds on those variables, d, we will need $(d+1)^n$ evaluations to prove that the polynomial is non-zero. This means that there is no deterministic algorithm that proves a polynomial is zero from its values, and degree bounds. Additional information is also needed.

For univariate polynomials over the reals, we can show that by choosing the points carefully, any polynomial with no more than T terms that vanishes at T points is identically zero.

Proposition 2. *Let $P(x)$ be a univariate polynomial with coefficients in* **Z**. *The number of positive real zeroes of $P(x)$ is less than or equal to* $\text{terms}(P) - 1$.

Proposition 2 follows immediately from Descartes' rule of signs since the maximum number of sign changes in the coefficients of $P(x)$ is $\text{terms}(P) - 1$. (For instance, Pólya and Szegö (1976) Part V, Chapter 1, problem 36.) The following corollary is merely a restatement of the proposition.

Corollary. *A univariate polynomial that vanishes at the integers $1, 2, \ldots, T$ is either identically zero or has more than T non-zero coefficients.*

Using some new techniques we show in section 5 that $O(nT^2)$ points suffice, where n is the number of variables in P. This is accomplished by finding a specialization of P to one variable that does not increase the number of non-zero terms and then applying Proposition 2. The previous best results were that $(d+1)^n+1$ sufficed, which is optimal for dense polynomials, but can be exponentially bad for sparse polynomials. In section 6 we give Ben Or and Tiwari's result that T suffices. In light of proposition 1 this is the best possible.

We occasionally use the notation p_i to indicate the ith prime. It is is also used to represent the ith element of the vector $\vec{p}$. Our intent should be clear from the context. Later, we will need a crude estimate for the size of the product of the first N primes. For our work we can use the crude estimate of

$$p_N \leq (N+1)^{1+\epsilon},$$

for some small constant ϵ. This is much weaker than the best known results, for instance Rosser and Schoenfeld (1962).

By applying Sterling's formula to the product of the first N primes we have

$$\begin{aligned}\log(p_1 p_2 \cdots p_N) &= \log(N+1)!^{1+\epsilon} \\ &= (1+\epsilon)\left(N+\frac{3}{2}\right)\log(N+2) + O(N)\end{aligned}$$

This proves the following proposition:

Proposition 3. *There exists a constant c_1 such that*

$$\log(p_1 p_2 \cdots p_N) \leq c_1 N \log N.$$

Many of the algorithms developed in this paper depend upon the special properties of Vandermonde matrices, which we summarize here. A *Vandermonde matrix* is a matrix of the form

$$V_n = \begin{pmatrix} 1 & k_1 & k_1^2 & \cdots & k_1^{n-1} \\ 1 & k_2 & k_2^2 & \cdots & k_2^{n-1} \\ \vdots & & \vdots & & \vdots \\ 1 & k_n & k_n^2 & \cdots & k_n^{n-1} \end{pmatrix},$$

where the k_i are chosen from some field. Similarly, a system of linear equations of the form

$$\begin{aligned} X_1 + k_1 X_2 + k_1^2 X_3 + \cdots + k_1^{n-1} X_n &= w_1 \\ X_1 + k_2 X_2 + k_2^2 X_3 + \cdots + k_2^{n-1} X_n &= w_2 \\ &\vdots \\ X_1 + k_n X_2 + k_n^2 X_3 + \cdots + k_n^{n-1} X_n &= w_n \end{aligned}$$

will be called a *Vandermonde system of equations.*

A matrix where the degrees of each row rise monotonically, but not necessarily linearly, is called a *generalized Vandermonde matrix*, viz,

$$\begin{pmatrix} k_1^{e_1} & k_1^{e_2} & k_1^{e_3} & \cdots & k_1^{e_n} \\ k_2^{e_1} & k_2^{e_2} & k_2^{e_3} & \cdots & k_2^{e_n} \\ \vdots & & \vdots & & \vdots \\ k_n^{e_1} & k_n^{e_2} & k_n^{e_3} & \cdots & k_n^{e_n} \end{pmatrix}.$$

The following well known theorem gives the determinant of a Vandermonde matrix.

Proposition 4. *The determinant of the Vandermonde matrix is*

$$\det V_n = \prod_{1 \le i < j \le n} (k_j - k_i).$$

As an immediate consequence the determinant of a Vandermonde matrix is nonzero if and only if the k_i are distinct.

A similar result is true for generalized Vandermonde matrices over the reals, but the proof is a little trickier. Notice that while proposition 4 applied to Vandermonde matrices over any field, the following proposition is only valid over the reals, which has characteristic zero. We know of no similar result for fields of positive characteristic.

Proposition 5. *The determinant of a generalized Vandermonde matrix is nonzero if the k_i are distinct positive real numbers.*

A proof of this result can be found in Gantmacher (1959), volume II, page 99.

The inverse of a Vandermonde matrix can be computed by the following well known technique. (See Press (1986), for example.) Multiply a Vandermonde matrix by a general n by n matrix:

$$\begin{pmatrix} 1 & k_1 & k_1^2 & \cdots & k_1^{n-1} \\ 1 & k_2 & k_2^2 & \cdots & k_2^{n-1} \\ \vdots & & \vdots & & \vdots \\ 1 & k_n & k_n^2 & \cdots & k_n^{n-1} \end{pmatrix} \cdot \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & & \vdots & & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix}$$

The jth element of the top row of the product of these two matrices is

$$a_{1j} + a_{2j}k_1 + a_{3j}k_1^2 + \cdots + a_{nj}k_1^{n-1} = P_j(k_1).$$

In fact the product above is

$$\begin{pmatrix} P_1(k_1) & P_2(k_1) & \cdots & P_n(k_1) \\ P_1(k_2) & P_2(k_2) & \cdots & P_n(k_2) \\ \vdots & \vdots & & \vdots \\ P_1(k_n) & P_2(k_n) & \cdots & P_n(k_n) \end{pmatrix}.$$

Choosing the $P_j(Z)$ to be

$$P_j(Z) = \prod_{\substack{i \ne j \\ 1 \le i \le n}} \frac{Z - k_i}{k_j - k_i},$$

we see that the product matrix is the identity, and thus the coefficients of the P_j are the columns of the inverse of the Vandermonde matrix. Each of the $P_j(Z)$ can be computed in $O(n)$ operations from a master polynomial, which itself can be computed in $O(n^2)$ operations. Thus the Vandermonde matrix can be inverted in $O(n^2)$ time.

Assume we wish to solve a Vandermonde system of equations like the following:

$$\begin{aligned} X_1 + k_1X_2 + k_1^2X_3 + \cdots + k_1^{n-1}X_n &= w_1 \\ X_1 + k_2X_2 + k_2^2X_3 + \cdots + k_2^{n-1}X_n &= w_2 \\ &\vdots \\ X_1 + k_nX_2 + k_n^2X_3 + \cdots + k_n^{n-1}X_n &= w_n \end{aligned} \tag{1}$$

If recognized as a Vandermonde system, these equations need only consume $O(n)$ space. They can be solved using $O(n)$ space by the following device.

Define

$$P(Z) = \prod_{1\le i\le n} (Z - k_i).$$

This polynomial contains $n + 1$ terms. The coefficient of Z^n is always 1. The polynomials $P(Z)/(Z - k_j)$ can be computed by synthetic division. It is the numerator of $P_j(Z)$. The value of $P(Z)/(Z - k_j)$ at k_j is the denominator of $P_j(Z)$. Thus each of the $P_j(Z)$ can be computed using $O(n)$ space and time. The computation of the X_i is arranged as follows.

$$\begin{pmatrix} X_1 \\ \vdots \\ X_n \end{pmatrix} = \begin{pmatrix} w_1 \cdot \operatorname{coef}(P_1, Z^0) \\ \cdots \\ w_1 \cdot \operatorname{coef}(P_1, Z^{n-1}) \end{pmatrix} + \cdots + \begin{pmatrix} w_n \cdot \operatorname{coef}(P_n, Z^0) \\ \cdots \\ w_n \cdot \operatorname{coef}(P_n, Z^{n-1}) \end{pmatrix}.$$

After each column vector on the right hand side is computed, it is added to the accumulating X_i and its storage may be reused by the following column vector.

This approach can also be applied to transposed Vandermonde systems like following

$$\begin{aligned} X_1 + X_2 + X_3 + \cdots + X_n &= w_1 \\ k_1X_1 + k_2X_2 + k_3X_3 + \cdots + k_nX_n &= w_2 \\ &\vdots \\ k_1^{n-1}X_1 + k_2^{n-1}X_2 + k_3^{n-1}X_3 + \cdots + k_n^{n-1}X_n &= w_2 \end{aligned} \tag{2}$$

since the inverse of the transpose of a matrix is the transpose of the inverse, we have the following formula for each of the X_i

$$X_i = w_1 \cdot \operatorname{coef}(P_1, Z^0) + w_2 \cdot \operatorname{coef}(P_1, Z^1) + \cdots + w_n \cdot \operatorname{coef}(P_1, Z^{n-1}).$$

These results are summarized in the following proposition.

Proposition 6. *The Vandermonde system (1) and the transposed Vandermonde system (2) over the field F can be solved in $O(n^2)$ operations over F. Furthermore, the space required is that of $O(n)$ elements of F. If $F = \mathbf{Q}$ and $K = \max|\operatorname{num} k_i| + \max|\operatorname{den} k_i|$ then the largest number used will require $O(n \log K)$ bits and in total $O(n^2 \log K)$ bits of storage will be required.*

3. Dense Interpolation

The general problem we consider in this paper is computing a polynomial from its values at certain points, whose choice may be part of some higher level algorithm. These polynomials may be multivariate and their coefficients generally lie in the rational integers, though occasionally they lie in a finite field. Many of these results can be extended to more complex fields, but we do not do this here. In this section we assume the number of variables in the polynomial is given, as well as degree bounds, but no additional assumptions are made. In particular, nothing is known about the number of non-zero terms in the polynomial.

3.1 Univariate Dense Interpolation

The simplest form of this problem consists of determining a univariate polynomial from its values at selected points. The straight forward approach works surprisingly well. Let

$$P(Z) = p_0 + p_1 Z + \cdots + p_{n-1} Z^{n-1} + p_n Z^n$$

be the polynomial to be determined. Assume the coefficients are over a field F, and let $z_0, \ldots, z_n$ be the set of distinct evaluation points. From the values of $P(z_i) = w_0$ we get the following system of linear equations in the unknown p_j.

$$\begin{aligned}
p_0 + p_1 z_0 + p_2 z_0^2 + \cdots + p_n z_0^n &= w_0 \\
p_0 + p_1 z_1 + p_2 z_1^2 + \cdots + p_n z_1^n &= w_1 \\
&\vdots \\
p_0 + p_1 z_n + p_2 z_n^2 + \cdots + p_n z_n^n &= w_n
\end{aligned}$$

This is a Vandermonde system and can solved quickly using the algorithm of Proposition 6.

3.2 Multivariate Dense Interpolation

As pointed out in the previous section, a polynomial in one variable of degree d can be determined from its values at $d+1$ points using $O(d^2)$ arithmetic operations. This result can be extended to multivariate problems.

Let $P(\vec{X})$ be the polynomial to be determined. It is a polynomial in n variables, $X_1, \ldots, X_n$, whose coefficients lie in an integral domain R. Each X_i appears to degree no more than d_i in P. Let $\ell = (d_1+1)(d_2+1)\cdots$, the maximum number of terms in P. Writing P as a sum of monomials using the vector notation we have

$$P(\vec{X}) = c_1\vec{X}^{\vec{e}_1} + c_2\vec{X}^{\vec{e}_2} + \cdots + c_\ell\vec{X}^{\vec{e}_\ell},$$

where the $\vec{e}_i$ run through each possible exponent combination. Choosing ℓ random n-tuples $\vec{x}_i$ and computing the values of $P(\vec{x}_i)$ gives a system of ℓ linear equations. In general, this requires $O(\ell^3)$ operations to solve, and perhaps more important $O(\ell^2)$ space.

There remains the question of solvability of the system of equations. Let M be an $n \times n$ matrix over a field F. M will be singular if and only if $\det M = 0$. Thus the singular matrices form an algebraic set of codimension 1 in the space of all $n \times n$ matrices. Thus the probability that the system of equations is singular is about $1/\#(F)$. For probabilistic algorithms this suffices. For deterministic algorithms more analysis is required. This is done in later sections by choosing the evaluation points carefully.

A recursive technique was used by Brown (1971) in the modular GCD algorithm was first used to bring the time requirements for interpolation down to $O(\ell^2)$. In this paper we use another approach that leads more naturally to the techniques for dealing with sparse polynomials.

Choose a random n-tuple $\vec{p}$. This is the initial evaluation point. Denote the values of the monomials $\vec{p}^{\vec{e}_i}$ by m_i. Additional evaluation points are obtained by raising $\vec{p}$ to successive powers (starting with 0). Notice that $(\vec{p}^j)^{\vec{e}_i} = m_i^j$. Thus we have the following system of equations to solve.

$$\begin{aligned}
c_1 + c_2 + \cdots + c_\ell &= P(\vec{p}^0)\\
c_1 m_1 + c_2 m_2 + \cdots + c_\ell m_\ell &= P(\vec{p})\\
c_1 m_1^2 + c_2 m_2^2 + \cdots + c_\ell m_\ell^2 &= P(\vec{p}^2)\\
&\vdots\\
c_1 m_1^{\ell-1} + c_2 m_2^{\ell-1} + \cdots + c_\ell m_\ell^{\ell-1} &= P(\vec{p}^{\ell-1})
\end{aligned}$$

This is the transpose of a Vandermonde system. As discussed in section 2, this system can be solved in $O(\ell^2)$ time and $O(\ell)$ space.

The key issue in this approach is guaranteeing the m_i are distinct so that the Vandermonde system will be non-singular. If the coefficient domain, R, is a unique factorization domain we can do this easily. For instance, assume R is the rational integers. We choose the components of $\vec{p}$ to be distinct primes, viz., $\vec{p} = (2, 3, 5, \ldots)$. By unique factorization each of the m_i will be distinct.

If the coefficient domain is a finite field $\mathbf{F}_q$, then the problem can be more difficult. The finite field must contain at least ℓ elements for the Vandermonde system to be non-singular. For the dense interpolation technique being discussed, the maximum value of ℓ is $(d+1)^n$. When $q < (d+1)^n$ the modular interpolation technique can still be used but elements should be chosen from an algebraic extension of $\mathbf{F}_q$ that has more than $(d+1)^n$ elements. In general, we can solve the system of equations using conventional $(O(\ell^3))$ techniques.

If the characteristic of $\mathbf{F}_q$ is sufficiently large, we can do better. Choose the components of $\vec{p}$ to be the rational primes, $(2, 3, 5, \ldots)$. If each of the m_i, when computed in $\mathbf{Z}$, is less than the characteristic of $\mathbf{F}_q$ then they will be distinct as elements of $\mathbf{F}_q$. For this to be the case the characteristic must be greater than

$$(2 \cdot 3 \cdots p_n)^d \approx n^{c_1 n d},$$

by proposition 3.

This idea of substituting primes for each of the variables was first suggested by Grigoriev and Karpinksi (1987), who were studying a problem involving polynomials with 0/1 coefficients. These ideas were first applied to interpolation by Tiwari (1987).

In the following paragraphs we analyze the hard case: We assume that q is greater than $(d+1)^n$, but that the characteristic of $\mathbf{F}_q$ is less than $n^{c_1 n d}$. The actual analysis is staightforward but somewhat lengthy. We consider the following somewhat more general question since its solution will be of use in analyzing the sparse algorithms. Let $\{\vec{e}_i\}$ be a set of T n-tuples where each component is bounded by d. (In the current case $T = (d+1)^n$.) What is the probability that for a randomly chosen $\vec{x} \in \mathbf{F}_q^n$ there is an $\vec{e}_i$ and $\vec{e}_j$ such that $\vec{x}^{\vec{e}_i} = \vec{x}^{\vec{e}_j}$?

We begin with an elementary enumeration proposition. The one dimensional version can be found in almost any book on elementary number theory.

Proposition 7. *Let $\vec{a} \neq 0$ be a fixed n-tuple where each component is an element of $\mathbf{Z}/(m)$ and c be the common GCD of the a_i and m. Let $\vec{x}$ be an n-tuple whose components range over $\mathbf{Z}/(m)$. Then $\vec{a} \cdot \vec{x}$ takes on m/c distinct values. These values divide the different $\vec{x}$ into m/c classes each containing cm^{n-1} different $\vec{x}$.*

Proof: First we reduce to the case where $c = 1$. Since $\vec{a} \cdot \vec{x}$ is a multiple of c for every $\vec{x}$, $\vec{a} \cdot \vec{x}$ can take on no more than m/c values, i.e. $0, c, 2c, \ldots$. Let αc be one of these values. Each solution of

$$\frac{\vec{a} \cdot \vec{x}}{c} \equiv \alpha \pmod{m/c} \tag{3}$$

gives rise to c^n solutions of $\vec{a} \cdot \vec{x} \equiv c\alpha \pmod{m}$. Thus if we can show that (3) has $(m/c)^{n-1}$ solutions, we are finished. The rest of the proof proceeds via a slightly complicated induction.

Consider the one dimensional case, $ax \equiv b \pmod{m}$. Since a and m are relatively prime, there is exactly one value of x that satisfies the relation for every value of b, as required by the proposition.

Now assume the proposition is true for all vectors $\vec{a}$ of dimension less than n. Let α be an arbitrary element of $\mathbf{Z}/(m)$. We want to show that $\vec{a} \cdot \vec{x} \equiv \alpha \pmod{m}$ has m^{n-1} zeroes. Without loss of generality we can assume that a_1 is not zero. If a_1 and m are relatively prime then for every choice of $a_2, \ldots, a_n$ there is a unique a_1 that satisfies the relation. Thus there are m^{n-1} zeroes of the relation as desired.

Assume that a_1 and m have a GCD of g. The relation has no zeroes if g does not divide $a_2x_2 + \cdots a_nx_n - \alpha$. Thus we consider the number of zeroes of

$$a_2x_2 + \cdots a_nx_n \equiv \alpha \pmod{g}.$$

Notice that $a_2, \ldots, a_n$ cannot have a GCD dividing g. Thus this equation has g^{n-2} zeroes modulo g. Each is the image of $(m/g)^{n-1}$ elements modulo m. Thus there are m^{n-1}/g choices of $a_2, \ldots, a_n$. Each one will give rise to g choices for x_1 giving the desired result. □

Corollary. *Let $\vec{a}$, m and c be as in the previous proposition. Then there are cm^{n-1} distinct solutions to $\vec{a} \cdot \vec{x} = 0 \pmod{m}$.*

Proof: 0 is always one of the values of $\vec{a} \cdot \vec{x}$ since $\vec{x}$'s components could be all zeroes. □

This result can now be used to answer the question raised above.

Proposition 8. *Let $\vec{e}_1, \ldots, \vec{e}_T$ be n-tuples where each component is less than d. There exists no more than*

$$\frac{d \cdot T \cdot (T-1) \cdot (q-1)^{n-1}}{2}$$

n-tuples $\vec{X}$ with components in F_q such that for some i and j $\vec{X}^{\vec{e}_i}$ and $\vec{X}^{\vec{e}_j}$ have the same values. Equivalently, for at least

$$(q-1)^{n-1}\left(q-1-\frac{d \cdot T \cdot (T-1)}{2}\right)$$

n-tuples $\vec{X}$, $\vec{X}^{\vec{e}_i}$ takes on distinct values.

Proof: Let g be a generator of the multiplicative group $\mathbf{F}_q^*$. Then for each n-tuple $\vec{X}$ we can assign another n-tuple $\vec{a}$ such that $X_i = g^{a_i}$, assuming no X_i is zero. The a_i are elements of $\mathbf{Z}/(q-1)$. Two monomials $\vec{X}^{\vec{e}_i}$ and $\vec{X}^{\vec{e}_j}$ have the same value when

$$\vec{x}^{\vec{e}_i} = g^{\vec{a} \cdot \vec{e}_i} = g^{\vec{a} \cdot \vec{e}_j} = vecx^{\vec{e}_j}.$$

That is, when $\vec{a} \cdot (\vec{e}_i - \vec{e}_j) = 0 \pmod{q-1}$. By the previous corollary there are $c(q-1)^{n-1}$ such zeroes, where c is the GCD of the elements of $\vec{e}_i - \vec{e}_j$ and $q-1$. Since $c \leq d$ there are at most $d(q-1)^{n-1}$ tuples $\vec{x}$ that cause these two terms to take on the same value.

There are $T(T-1)/2$ distinct pairs of $\vec{e}_i$, so the maximum number of $\vec{x}$ that cause a pair of $\vec{x}^{\vec{e}_i}$ to take on the same value is

$$\frac{d \cdot T \cdot (T-1) \cdot (q-1)^{n-1}}{2}.$$

□

Since there are only $(q-1)^{n-1}$ possible $\vec{x}$ (ignoring those with a zero component), we have the following corollary.

Corollary. *The probability that a randomly chosen $\vec{x}$ will cause two of the $\vec{x}^{\vec{e}_i}$ to have the same value is*

$$\frac{d \cdot T \cdot (T-1)}{2(q-1)}.$$

If we wish the probability of a collision to be less than ϵ, then for dense polynomials this means that

$$q > \frac{(d+1)^{2v+1}}{2\epsilon}.$$

This is actually quite impractical for polynomials with large numbers of variables and high degree. Fortunately, many problems are sparse, i.e. $T \ll (d+1)^v$, which gives much better results. This is the topic of the next section.

4. Sparse Interpolation

The purpose of this section is to develop Zippel's sparse interpolation algorithm, which gives a probabilistic resolution of the interpolation problem. What is presented is an improvement of the author's original results based on some of the ideas first suggested by Ben-Or and Tiwari. This algorithm is given no information about the number of non-zero terms in the polynomial being interpolated. Instead it develops an estimate of the number of terms as each new variable is introduced. As a consequence its performance depends upon the actual number of non-zero terms in the polynomial rather than an *a priori* bound. This probabilistic algorithm tends to be more useful in practical situations than the deterministic algorithms presented in the following sections.

This section has been divided into three subsections. In the first we give a demonstration of the algorithm and its benefits. In the subsequent subsections we give a more formal presentation of its details, and analyze the algorithm's performance.

4.1 Heuristic Presentation

As before we wish to determine the polynomial $P(\vec{X}) \in U[\vec{X}]$ from its values, where U is a field with sufficiently many distinct elements. We assume that d_i bounds the degree of X_i in P. The sparse interpolation algorithm computes P one variable at a time. That is, we initially compute $P(a_1, a_2, \ldots, a_n)$, then $P(X_1, a_2, \ldots, a_n)$, then $P(X_1, X_2, a_3, \ldots, a_n)$ and so on, until we have determined $P(\vec{X})$. The introduction of each new variable is called a *stage* of the algorithm. We use clues from the polynomial produced in the preceding stage to minimize the effort required to produce the next polynomial in the sequence.

The description of the sparse interpolation algorithm becomes rather involved and it is easy to get bogged down by all the subscripts and variables involved, but it is fundamentally quite simple. In this section we give an explicit example.

Assume we wish to interpolate a polynomial in three variables, $P(X, Y, Z)$ over a field, where the degree of each variable is not greater than 5. When the polynomial is dense, there are 125 different coefficients that must be determined. We assume that most of these coefficients are zero and that P possesses only a few non-zero monomials. By using one of the dense interpolation schemes of section 3, we can compute $P(X, y_0, z_0)$ from $P(x_0, y_0, z_0), P(x_1, y_0, z_0), \ldots, P(x_5, y_0, z_0)$. Assume this yields

$$P(X, y_0, z_0) = c_0 X^5 + c_1 X + c_2.$$

This is the end of the first stage.

Beginning stage two, we know that $P(X, Y, Z)$ can be written as

$$P(X, Y, Z) = P_5(Y, Z) X^5 + P_4(Y, Z) X^4 + \cdots + P_0(Y, Z).$$

From the first interpolation we know that $P_5(y_0, z_0) = c_0$, $P_1(y_0, z_0) = c_1$ and $P_0(y_0, z_0) = c_2$. Since the other coefficients are zero

$$P_4(y_0, z_0) = P_3(y_0, z_0) = P_2(y_0, z_0) = 0.$$

The key step in the sparse interpolation algorithm is to assume that this is true for all values of Y and Z. That is, that

$$P_4(Y, Z) = P_3(Y, Z) = P_2(Y, Z) = 0.$$

In typical calculations, where y_0 and z_0 are chosen at random from a large set of possibilities, this is a good assumption. Proposition 9 below gives a precise measure of how good an assumption this is.

We now choose a new value for Y, y_1, and compute $P(X, y_1, z_0)$. Without the assumption of the previous paragraph, this interpolation would require 6 additional values of P. Instead we assume that $P(X, y_1, z_0)$ contains only 3 non-zero terms, i.e.,

$$P(X, y_1, z_0) = c_3X^5 + c_4X + c_5,$$

where the c_3, c_4 and c_5 are to be determined. Since there are only three unknowns to be determined only three new values of P are required.

This process is repeated until we have six polynomials.

$$\begin{aligned} c_0X^5 + c_1X + c_2 &= P(X, y_0, z_0) \\ c_3X^5 + c_4X + c_5 &= P(X, y_1, z_0) \\ &\vdots \\ c_{15}X^5 + c_{16}X + c_{17} &= P(X, y_5, z_0) \end{aligned}$$

By the dense interpolation algorithm of section 3, the coefficients of the X^5 terms can be interpolated to produce a polynomial in Y, and similarly for the linear and constant terms. Combining these results we have $P(X, Y, z_0)$. Notice that we have only needed $6 + 3 + 3 + 3 + 3 + 3$ values of P to compute this polynomial. The dense interpolation scheme would have required almost twice as many evaluation points.

Beginning the third stage, let us assume this gives the polynomial

$$\begin{aligned} P(X, Y, z_0) &= k_1X^5 + (k_2Y^4 + k_3Y)X + k_4Y^5 \\ &= k_1X^5 + k_2XY^4 + k_3XY + k_4Y^5, \end{aligned}$$

where the k_i are elements of the ground field. We are now in a position to begin the process again, but this time introducing the variable Z. To do this we need to calculate the polynomials $P(X, Y, z_0), P(X, Y, z_1), \ldots, P(X, Y, z_5)$. We assume that those XY-monomials that did not arise in $P(X, Y, z_0)$ have coefficients of zero in $P(X, Y, Z)$.

Thus to compute

$$P(X, Y, z_1) = k_5X^5 + k_6XY^4 + k_7XY + k_8Y^5$$

we only need interpolate four values of P. Thus the additional 5 polynomials only required $5 \times 4 = 20$ evaluation points. Without the sparsity assumptions each of the 5 polynomial would have required 36 evaluation points, and 180 in all.

4.2 Formal Presentation

To fix our notation, assume we want to use the sparse interpolation algorithm to determine a polynomial $P(X_1, \ldots, X_n) \in F[\vec{X}]$ where we know that each X_i does not appear to degree higher than d and that there are t non-zero monomials in P. Furthermore, we assume that we can compute the value of P given a value for $\vec{X}$. It is convenient to consider just one stage of the interpolation. The computation of $P(\vec{X})$ being just a sequence of n stages.

Now assume that we have performed the first $k-1$ stages of the sparse modular algorithm and we are about to begin the kth stage. From the previous stage's computation we have

$$P(X_1, \dots, X_{k-1}, x_{k0}, \dots, x_{n0}) = p_{10}\vec{X}^{\vec{e}_1} + p_{20}\vec{X}^{\vec{e}_2} + \cdots + p_{T0}\vec{X}^{\vec{e}_T}.$$

The set of exponents of $P(X_1, \dots, X_{k-1}, x_{k0}, \dots, x_{n0})$ is called the *skeleton* of P, which we denote by $\operatorname{skel} P$. Since there are t non-zero monomials in P, the skeleton of P can never have more than t elements.

Throughout this stage, the values assigned to $X_{k+1}, \dots, X_n$ do not vary. To simplify the notation, we will omit them.[1] We write

$$P'(y_1, \dots, y_k) = P(y_1, \dots, y_k, x_{k+1,0}, \dots, x_{n0}).$$

The computation of $P(X_1, \dots, X_{k-1}, X_k)$ proceeds in two phases. In the first we determine

$$P'(X_1, \dots, X_{k-1}, x_{kj}) = p_{1j}\vec{X}^{\vec{e}_1} + p_{2j}\vec{X}^{\vec{e}_2} + \cdots + p_{Tj}\vec{X}^{\vec{e}_T},$$

for $j = 0, \dots, d$ by the following technique:

For each of $d+1$ randomly chosen values of X_k, x_{kj} perform the following. Pick a random $k-1$-tuple denoted by $(y_1, \dots, y_{k-1}) = \vec{y}$, such that each of $\vec{y}^{\vec{e}_i}$ are distinct. Since the $\vec{e}_i$ are known, verifying that this is the case is easy. Actually finding $\vec{y}$ is discussed in the analysis of the next subsection. This value $\vec{y}$ allows us to set up the following (non-singular) system of linear equations

$$\begin{aligned}
P'(1, \dots, 1, x_{kj}) &= p_{1j} + p_{2j} + \cdots + p_{Tj} \\
P'(y_1, \dots, y_{k-1}, x_{kj}) &= p_{1j}\vec{y}^{\vec{e}_1} + p_{2j}\vec{y}^{\vec{e}_2} + \cdots + p_{Tj}\vec{y}^{\vec{e}_T} \\
P'(y_1^2, \dots, y_{k-1}^2, x_{kj}) &= p_{1j}\vec{y}^{2\vec{e}_1} + p_{2j}\vec{y}^{2\vec{e}_2} + \cdots + p_{Tj}\vec{y}^{2\vec{e}_T} \\
&\vdots \\
P'(y_1^T, \dots, y_{k-1}^T, x_{kj}) &= p_{1j}\vec{y}^{t\vec{e}_1} + p_{2j}\vec{y}^{t\vec{e}_2} + \cdots + p_{tj}\vec{y}^{t\vec{e}_t}
\end{aligned}$$

This is a Vandermonde system of equations and can be solved by the techniques of Section 2 in $O(t^2)$ time while requiring only $O(t)$ space. The result will be a polynomial

$$P(X_1, \dots, X_{k-1}, x_{kj}, x_{k+1,0}, \dots, x_{n0}),$$

[1] In practical implementations this may be more than notational. Eliminating the variables that do not vary at this stage can save significant time when computing the values of P.

for each of the $d+1$ values x_{kj}.

In the second phase, we independently interpolate the coefficients of each monomial, using the dense interpolation algorithm. The results of these interpolations can be combined to produce

$$P'(X_1, \ldots, X_{k-1}, X_k) = p_1(X_k)\vec{X}^{\vec{e}_1} + p_2(X_k)\vec{X}^{\vec{e}_2} + \cdots + p_T(X_k)\vec{X}^{\vec{e}_T}.$$

The dense interpolation yielded the univariate polynomials $p_i(X_k)$. This polynomial is in turn expanded to get

$$P(X_1, \ldots, X_k, x_{k+1,0}, \ldots, x_{n0}) = p'_{10}\vec{X}^{\vec{e}'_1} + p'_{20}\vec{X}^{\vec{e}'_2} + \cdots + p'_{T0}\vec{X}^{\vec{e}'_T},$$

and we are ready to begin lifting the next variable.

4.3 Analysis

We begin by presenting the probabilistic resolution of the zero avoidance problem. The following proposition gives a sharp estimate of how difficult it is to avoid the zeroes of a polynomial given only degree bounds.

Proposition 9. *Let k be a field, f any element of $k[X_1, \ldots, X_n]$ such that the degree of X_i in f is bounded by d_i. Let $Z_n(B)$ be the number of zeroes of f, $\vec{x}$ such that $x_i \in S$ (a set with B elements $B \gg d_i$). Then*

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

Historical note: This result initially appeared in two papers simultaneously and independently at the EUROSAM '79 conference in Marseille during the summer of 1979. Schwartz gave the second estimate of this proposition while Zippel gave a version of the first. The proof given here is a simplification and extension of that given in Zippel (1979).

Proof: There are at most d_n values of X_n at which f is identically zero. So for any of these d_n values of X_n and any value for the other X_i, f is zero. This comes to d_nB^{n-1}. For all other $B - d_n$ values of X_n we have a polynomial in $n-1$ variables. The polynomial can have no more than $Z_{n-1}(B)$ zeroes. Therefore,

$$Z_n(B) \le d_nB^{n-1} + (B - d_n)Z_{n-1}(B).$$

Rather than solving this recurrence for Z_n, we solve it for $N_n = B^n - Z_n$. Since Z_1 is less than or equal to d_1, $N_1 \ge (B - d_1)$. This is the basis step of the inductive proof. Writing the recurrence in terms of N_n we have

$$B^n - N_n(B) \le d_nB^{n-1} + (B - d_n)(B^{n-1} - N_{n-1}(B))$$

or

$$N_n(B) \geq (B - d_n)N_{n-1}(B),$$

from which the proposition follows. $\square$

Polynomials of the following actually have $B^n - (B - d_1) \cdots (B - d_n)$ zeroes with components less than B

$$f(X_1, \ldots, X_n) = \prod_{0 \leq i_1 \leq d_1} (X_1 - x_{1i_1}) \quad \cdots \prod_{0 \leq i_n \leq d_n} (X_n - x_{ni_n}).$$

Thus the inequality in the proposition cannot be further strengthened. The following corollary phrases this result as a probability.

Corollary. *Let $f_1, f_2, \ldots, f_s$ be elements of $k[X_1, \ldots, X_n]$, where the degree in each variable is bounded by d. Let $\mathcal{P}(f_1, \ldots, f_s)$ be the probability that a randomly chosen point $\vec{x}$ is a zero of any of the f_i, where x_i is an element of a set with B elements. Then*

$$\mathcal{P}(f_1, \ldots, f_s) < \frac{nds}{B}.$$

Proof: Let $f = f_1 f_2 \cdots f_s$. The degree of each variable in f is bounded by ds. Applying the previous proposition, we see that the number of zeroes of f is bounded by $ndsB^{n-1}$, for sufficiently large B. Since there are B^n possible $\vec{x}$ to choose from, we have the corollary. $\square$

This corollary gives the probabilistic solution to the zero avoidance problem. Let P be an element of $\mathbf{Z}[X_1, \ldots, X_n]$. Choose a random point in $\mathbf{Z}^n$ with components less than B. The probability that this point will be a zero of P is less than

$$\frac{nd}{B}.$$

Thus to keep the chance of error below ϵ, using a single evaluation, we must choose

$$B > \frac{nd}{\epsilon}.$$

Turning now to the sparse modular interpolation algorithm, if all the probabilistic assumptions hold, the cost of lifting a single variable can be computed as follows. In phase 1 we compute $d+1$ polynomials at a cost of at most t evaluation points each, requiring $O(dt^2)$ time and $O(dt)$ space. The dense interpolation in phase 2 requires $O(d^2)$ steps for each coefficient that is interpolated. At most t such interpolations are performed so a total of $O(td^2)$ steps are required. Since n stages need to be performed the total time requirements are $O(ndt(d+t))$, while the maximum space requirement is always $O(dt)$. Remember that t is the actual number of terms in P, not an a *priori* bound on the number of terms in P.

As we shall see, the chance for error in the interpolation depends entirely on the initial evaluation point $(x_{10}, x_{20}, \ldots, x_{n0})$. By performing several interpolations with different initial points we can decrease the chance of error. This may be appropriate in practical implementations. Here we use ϵ to denote the chance for error from a single starting point. We assume P is a polynomial over a finite field, $\mathbf{F}_q$. We want to determine ϵ as a function of q, d, n and t. In practical implementation, P will most likely be a polynomial over $\mathbf{Z}$. Then q is chosen to minimize ϵ and still remain efficient. To convert from a solution in a finite field to one over the integers a coefficient bound is needed for the solution in $\mathbf{Z}$. In this section we ignore these issues. From a theoretical point of view, we continue to compute in $\mathbf{Z}$ (or $\mathbf{Q}$ as necessary), and restrict our random choices to integers less than q.

There are two sources of potential error in this algorithm. First, the structure inherited from earlier stages in phase 1 structure may be incorrect. That is, a term that was assumed to be zero really wasn't zero. To be precise, consider a three variable problem. Assume the polynomial to be computed is

$$p_1(Y,Z)X^{e_1} + p_2(Y,Z)X^{e_2} + \cdots + p_t(Y,Z)X^{e_t},$$

and the initial evaluation point is (x_0, y_0, z_0). After the single variable interpolation computed in stage one we have the polynomial

$$p_1(y_0,z_0)X^{e_1} + p_2(y_0,z_0)X^{e_2} + \cdots + p_t(y_0,z_0)X^{e_t}.$$

In passing from the one variable case (X) to the two variable case $(X,\ Y)$, the algorithm just presented assumes the structure given above is correct. If p_i vanished at (y_0, z_0) we would have assumed e_i was zero erroneously. At the end of this stage we will have the bivariate polynomial

$$q_1(z_0)(X,Y)^{\vec{f}_1} + q_1(z_0)(X,Y)^{\vec{f}_2} + \cdots + q_t(z_0)(X,Y)^{\vec{f}_t}.$$

Again, if any of the q_i vanish at z_0 we will get erroneous results. To compute the exponent vectors correctly, we need to assume that the p_i and q_i do not vanish at the initial point (x_0, y_0, z_0). These are the polynomials whose zeroes we must avoid.

At the ith stage of the interpolation process, there are at most t polynomials in $n-i$ variables whose zeroes must be avoided. The aggregate number of non-zero terms these polynomials contain must be less than t. The degrees of each polynomial is bounded by d. So by proposition 9, the chance of the initial evaluation point being the zero of any of these is

$$\frac{n^2 dt}{q}.$$

Now consider the probability that the Vandermonde systems are singular. These systems are of rank at most t. By the corollary to proposition 8, the probability that this system is singular is

$$\frac{d \cdot t \cdot (t-1)}{2(q-1)} \approx \frac{dt^2}{2q}.$$

At each stage there are d such systems to be solved and there are $n-1$ stages in all, so the probability that one of them will fail is bounded by

$$\frac{nd^2t^2}{q}.$$

Thus we have the following proposition.

Proposition 10. *Let P be a polynomial in n variables, each of degree no more than d and with t ($\gg n$) non-zero terms. Assume the coefficients of P lie in a finite field with q elements. The probability that the sparse interpolation algorithm will give the wrong answer for this polynomial is less than*

$$\frac{nd^2t^2}{q}.$$

The randomly chosen values must be chosen from a set of at least

$$\frac{nd^2t^2}{\epsilon}$$

values for the probability of error to be less than ϵ.

Since we cannot know the true number of non-zero terms of P before beginning the algorithm, the random values must be chosen from a set of

$$\frac{nd^2T^2}{\epsilon}$$

points.

5. Deterministic Zero Avoidance

As mentioned earlier, proposition 2 shows that univariate polynomials over the rational integers cannot have many real zeroes. We extend this proposition to one for a multivariate polynomial in the variables X_i by finding a substitution $(X_i \mapsto Z^{e_i})$ that sends a multivariate polynomial into a univariate polynomial. We then apply the proposition to the univariate polynomial to get our result. The crucial part of the proof is to show that we can find a substitution such that $P(Z^{\vec{e}})$ is not identically zero.

Before, proceeding with our version of the bounds, it is instructive to examine the bound derivable from Kronecker's technique, van der Waerden (1953). We are given a polynomial in n variables, $P(X_1, \ldots, X_n)$ where the maximum degree in any one variable is d and assume there are no more than T non-zero terms in P. Let ℓ be an integer larger than d. Consider the substitution, $X_i \mapsto Z^{\ell^{i-1}}$. A monomial $\vec{X}^{\vec{e}}$ is mapped to a monomial in Z raised to the power:

$$e_1 + e_2\ell + e_3\ell^2 + \cdots + e_n\ell^{n-1}.$$

Since each of the e_i are strictly smaller than ℓ this mapping is one to one and invertible.

Furthermore, we haven't changed the number of non-zero terms in the polynomial, i.e. terms($P(\vec{X})$) = terms($P(Z)$). By proposition 1, if $P(Z)$ vanishes for T positive values of Z, then it (and thus $P(\vec{X})$) is identically zero. This would be our desired proposition if the values chosen for the X_i were small enough. The smallest integer values we can choose for Z are $1, 2, \ldots, T$. Thus the values for X_i are

$$1^{\ell^{i-1}}, 2^{\ell^{i-1}}, \ldots, T^{\ell^{i-1}}.$$

Unfortunately, the size of the largest substitution, $T^{\ell^{n-1}}$ is exponential in the number of variables.

This basic idea can be salvaged by a more flexible choice of exponent substitutions. Rather than using an invertible substitution, as Kronecker does, we choose one that merely guarantees that $P(Z)$ is not identically zero if $P(\vec{X})$ is not. In light of the results of Ben-Or and Tiwari the importance of this result is somewhat diminished. However the technique used to reduce a multivariate polynomial to a univariate polynomial, while preserving the number of non-zero terms seems quite powerful.

We begin with a definition and some lemmas.

Definition: 1. Let $\mathcal{A}$ be a set of n-tuples with components in a ring R. $\mathcal{A}$ is said to be *maximally independent* if every subset of n elements of $\mathcal{A}$ is R-linearly independent.

In our situation, each element of $\mathcal{A}$, $\vec{s}$, corresponds to a substitution $X_i \mapsto Z^{s_i}$. The following lemma shows that there exist sets of N maximally independent n-tuples with entries not much larger than N.

Lemma 1. *Let S be a positive integer, and p the smallest prime larger than S. There exists a maximally independent set of S n-tuples with components in* **Z** *where each of the components of the n-tuples is less than p.*

Proof: First we show that we can construct arbitrarily large maximally independent sets of n-tuples. Then by reducing them modulo a prime we get the n-tuples required by the lemma. Consider n-tuples of the form $(1, k, k^2, \ldots, k^{n-1})$. For n of these to be independent the determinant of the matrix

$$\begin{pmatrix} 1 & k_1 & k_1^2 & \cdots & k_1^{n-1} \\ 1 & k_2 & k_2^2 & \cdots & k_2^{n-1} \\ \vdots & & \vdots & & \vdots \\ 1 & k_n & k_n^2 & \cdots & k_n^{n-1} \end{pmatrix}$$

must not be zero. Since this is a Vandermonde matrix, its determinant is $\prod_{i>j}(k_i - k_j)$, by proposition 3.

Thus if the k_i are distinct the vectors they generate will be linearly independent. In particular if we let $\vec{u}_k = (1, k, \ldots, k^{n-1})$ then any subset of n of the $\vec{u}_k$ will be linearly independent. Furthermore, if we reduce the elements of $\vec{u}_k$ by a prime larger than any of the k, the n-tuples remain maximally independent. $\square$

Lemma 2. *Let $L(\vec{X}) = \vec{w} \cdot \vec{X}$ be a linear form in the n variables X_i that is not identically zero. If $\vec{p}_1, \ldots, \vec{p}_n$ are linearly independent n-vectors, then $L(\vec{p}_i) \neq 0$ for some i.*

Proof: Since the n-tuples $\vec{p}_i = (p_{i1}, p_{i2}, \ldots, p_{in})$ are linearly independent, the matrix

$$\mathbf{A} = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & & & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{pmatrix}$$

is non-singular. Denote by $\vec{w}$ the column vector $(w_1, \ldots, w_n)^T$. If L vanishes at each of the n-tuples $\vec{p}_i$ then

$$\mathbf{A} \cdot \vec{w} = 0.$$

Since $\mathbf{A}$ is non-singular, $\vec{w}$ must be identically zero. $\square$

Lemma 3. *Let $L_j(\vec{X}) = \vec{w}_j \cdot \vec{X}$ be a set of T linear forms in n variables X_i, where none of the forms is identically zero. There exists a set of $(n-1) \cdot T + 1$ n-tuples such that for one of these n-tuples none of the L_j vanish. Furthermore, the components of these n-tuples can be chosen such that each component is less than $2nT$.*

Proof: By the previous lemma, each L_i can vanish at no more than $n-1$ independent n-tuples. Assuming none of the forms vanish at the same n-tuple, there can only $(n-1) \cdot T$ n-tuples for which one of the forms vanish. $\square$

This Lemma can be extended somewhat to give a estimate of the number of n-tuples required to ensure that each linear form takes on a distinct value. This is important enough to justify calling it a proposition.

Proposition 11. *Let $L_j(\vec{X}) = \vec{w}_j \cdot \vec{X}$ be a set of T distinct linear forms in n variables X_i. There exists a set of*

$$\frac{(n-1)\cdot T\cdot(T-1)}{2}+1$$

n-tuples such that each L_j takes on a different value for at least one of them, and where the components are each less than $nT(T-1)$.

Proof: Consider the set of forms

$$M_{ij} = (\vec{w}_i - \vec{w}_j)\cdot\vec{X}.$$

Ignoring the diagonal forms (M_{ii}), which are identically zero, there are $T(T-1)/2$ distinct forms up to sign. L_i and L_j have the same value for some n-tuple, if and only if M_{ij} vanishes at the same n-tple. By Lemma 3, there exists a set of

$$\frac{(n-1)\cdot T\cdot(T-1)}{2}+1$$

n-tuples such that for one them, none of the M_{ij} vanish, and each has components less than $nT(T-1)$. $\square$

Proposition 12. *There exists a set of nT^2 n-tuples such that there is no polynomial with less than T non-zero terms that vanishes at each of the n-tuples. Furthermore, the absolute value of the components of the n-tuples is less than T^{2nT}, and they have size $O(nT\log T)$.*

This proposition is proven by applying the same type of reasoning used earlier with Kronecker's trick, using a sufficiently large, maximally independent set of n-tuples.

Proof: Assume $P(X_1,\ldots,X_n)$ is not identically zero and let the terms of P be

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

The substitution $X_i \mapsto Z^{u_i}$ sends this polynomial into

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

This substitution must be chosen so that $P(Z)$ is not identically zero. This can be done by requiring that for any $i \neq 1$

$$\vec{e}_1\cdot\vec{u} \neq \vec{e}_i\cdot\vec{u}$$

or equivalently $(\vec{e}_i - \vec{e}_1)\cdot\vec{u} \neq 0$.

By Lemma 3, we can choose a set of $(n-1)(T-1)+1$ maximally independent n-tuples such that one of them satisfies $(\vec{e}_i - \vec{e}_1) \cdot \vec{u} \neq 0$. We can bound the components of $\vec{u}$ by p where p be the smallest prime larger than $(n-1)(T-1)+1$. Notice that $p < 2nT$.

Each of the n-tuples gives rise to a mapping from $P(\vec{X})$ to $P(Z)$. Since $P(Z)$ has no more than T monomials, we need not try more than T positive integer values for each $\vec{u}$. In particular we can use the values $1, 2, \ldots, T$ for Z. Thus there exists a set of $(n-1)(T-1)T + T < nT^2$ points that satisfies the requirements of the theorem. Furthermore, each component of the substitution is bounded by $T^p < T^{2nT}$. □

Proposition 13. *There exists a set of nT^2 n-tuples, whose components are of size $O(nT \log T)$, such that for every set of polynomials $P_i \in \mathbf{Z}[x_1, \ldots, x_n]$ with*

$$\sum_i \text{terms}(P_i) < T,$$

there is at least one n-tuple where none of the polynomials vanish.

This proposition follows from proposition 12 and the observation that if $t_1 + \cdots + t_k = T$, then the maximum value of $t_1^2 + \cdots + t_n^2$ is T^2.

The remaining result in this section is due to Ben-Or and Tiwari (1988). By using a direct multivariate approach to the zero-avoidance problem, they improve the $O(nT^2)$ result of proposition 12 for the zero avoidance problem to T, which is best possible. Ben-Or and Tiwari's main idea for this problem is contained in the following proposition.

Proposition 14. *Let $P(\vec{X})$ be a non-zero polynomial in $R[\vec{X}]$ with at most T terms and with monomial exponent vectors $\vec{e}_i$. Assume there exists an n-tuple $\vec{x}$ such that the $\vec{x}^{\vec{e}_i}$ are distinct. Then not all of $P(\vec{x}^0), P(\vec{x}), P(\vec{x}^2), \ldots, P(\vec{x}^{T-1})$ are zero.*

Proof: Denote $\vec{x}^{\vec{e}_i}$ by m_i. By assumption, each of the m_i are distinct. If P vanished at each of the $\vec{x}^i$ then the following system of linear equations would hold.

$$\begin{aligned} c_1 + c_2 + \cdots c_T &= 0 \\ c_1 m_1 + c_2 m_2 + \cdots + c_T m_T &= 0 \\ c_1 m_1^2 + c_2 m_2^2 + \cdots + c_T m_T^2 &= 0 \\ &\vdots \\ c_1 m_1^{T-1} + c_2 m_2^{T-1} + \cdots + c_T m_T^{T-1} &= 0 \end{aligned}$$

Since this is a Vandermonde system and we have assumed that the m_i are distinct, the system of equations is non-singular. Thus the c_i must all be zero, and P must be identically zero for all of $P(\vec{x}), P(\vec{x}^2), P(\vec{x}^3), \ldots, P(\vec{x}^{T-1})$ to vanish. □

The key then is finding a substitution that keeps the monomials distinct. If P is a polynomial over a unique factorization domain (such as the rational integers) then this is relatively easy—we choose the components of $\vec{x}$ to be distinct primes. In this case each of the m_i must be distinct by unique factorization. For polynomials over finite fields estimates of the difficulty in finding such the right initial substitution can be made form proposition 7, but this leads to a probabilistic algorithm.

The following proposition considers the zero avoidance problem for several polynomials.

Proposition 15. *Let $P_1(\vec{X}), \ldots, P_s(\vec{X})$ be non-zero polynomials in $U[X_1, \ldots X_n]$, U a unique factorization domain and assume that* $\operatorname{terms} P_1 + \cdots + \operatorname{terms} P_s = T$. *Let $\vec{x}$ be a vector of n primes in U. Then for integer j, $0 \le j \le T$, all of $P_i(\vec{x}^j)$ are different from zero.*

Proof: Denote the points $\{\vec{x}^0, \vec{x}, \ldots, \vec{x}^T\}$ by $\mathcal{A}$. P_i cannot vanish at more than $\operatorname{terms} P_i$ elements of $\mathcal{A}$ by proposition 5. Since $\mathcal{A}$ contains $T+1$ points, there must be one for which none of the P_i vanish.

6. New Interpolation Algorithm

Using either of the deterministic solutions of the zero avoidance problem given in the previous section (propositions 13 and 15), it is possible to modify the probabilistic sparse polynomial interpolation algorithm of section 4 to make it deterministic.

As usual, we wish to interpolate a sparse polynomial with no more than T non-zero terms, $P(\vec{X}) \in F[X_1, \ldots, X_n]$, from its values. As in the last section we will only consider the case when F is the rational integers or a finite field of sufficiently large characteristic. For simplicity our discussion will use $F = \mathbf{Z}$. Thus we can guarantee that the Vandermonde systems of equations are always non-singular, by using as the initial starting point: $(2, 3, 5, \ldots, p_n)$, where p_n is the nth prime.

The only remaining source of erroneous answers in the probabilistic algorithm of section 4, is that coefficient polynomials may vanish at the starting point. To be more precise, assume the starting point of the interpolation is $x_{10}, x_{20}, \ldots, x_{n0}$. Consider stage k, where we are introducing X_k. We can write $P(\vec{X})$ as

$$P(\vec{X}) = p_{1k}(X_{k+1}, \ldots, X_n)(X_1, \ldots, X_k)^{\vec{e}_1} + \cdots \\ + p_{tk}(X_{k+1}, \ldots, X_n)(X_1, \ldots, X_k)^{\vec{e}_t}.$$

If the polynomials p_{ik} do not vanish at the starting point, then skeleton produced at stage k will be a correct image of $\operatorname{skel} P$. If this is the case we say the starting point is a *stage k good starting point.* If the starting point was not good, then the resulting skeleton will be strictly smaller than the correct one at that stage.

The deterministic version of the sparse modular algorithm assumes that at stage $k-1$, the polynomial it is given has the correct skeleton. It then produces a k variable polynomial that has the correct skeleton, by ensuring that it has used a starting point for which none of the p_{ik} vanish. This is easily done by performing the operations of stage k, T times, using $(x^s_{k+1,0}, \ldots, x^s_{n0})$ as the values for the undetermined variables. Since the total number of terms in p_{ik} not greater than T, by proposition 14, one of these starting points will be stage k good. Since we know the correct $k-1$ skeleton it is not necessary to repeat lower stages of the algorithm.

Thus this algorithm will require T times more operations than the probabilistic version. The components of the evaluation points are always primes (or a random integer for X_n). Thus the largest component will be p_n^T, whose size is approximately $O(T \log n)$.

7. Conclusions

We have presented new deterministic solutions to both the zero avoidance problem and the interpolation problem for sparse polynomials. The zero avoidance technique of proposition 13 reduces multivariate problems to univariate problems. The interpolation algorithm presented may have better performance than Ben-Or and Tiwari's interpolation algorithm if the bound on the number of terms is not sharp.

Unfortunately, these deterministic results do not immediately yield deterministic algorithms for the multivariate polynomial greatest common divisor (GCD) and factorization problems. For the GCD problem a technique for avoiding the zeroes of the resultant of the two polynomials is needed. Unfortunately, straight forward estimates of the number of terms of the resultant are exponential in the number of variables even if the original polynomials were sparse. For the factorization problem, using the current techniques, there still remains the need for an effective version of the Hilbert Irreducibility theorem with good constants. The existing versions give probabilistic results, von zur Gathen (1983), Heintz and Sieveking (1981) and Kaltofen (1985b).

8. Acknowledgements

This work benefited greatly by discussions with Barry Trager. The one dimensional version of these ideas, Proposition 2, was suggested by Gregory Chudnovsky. The key idea in Lemma 1 was suggested by Carlo Traverso. Prasoon Tiwari was very generous in clarifying his and Ben-Or's results. Two anonymous referees provided many valuable comments and suggestions. Their efforts have significantly improved the presentation.

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. Work done in the Laboratory for Computer Science was supported 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

Ben Or, M. and Tiwari, P. (1988). A Deterministic Algorithm for Sparse Multivariate Polynomial Interpolation. *20th Symposium on the Theory of Computing*, 301–309.

Brown, W. S. (1971). On Euclid's Algorithm and the Computation of Polynomial Greatest Common Divisors. *J. Assoc. Comp. Mach.* **18/4**, 478–504.

Gantmacher, F. R. (1959). *The Theory of Matrices*, Chelsea Publ., New York.

von zur Gathen, J. (1983). Factoring Sparse Multivariate Polynomials. *Proceedings, IEEE Symposium on the Foundations of Computer Science*, 172–179.

von zur Gathen, J. (1985). Irreducibility of Multivariate Polynomials. *J. of Computer and System Sciences* **31**, 225–264.

von zur Gathen, J., Kaltofen, E. (1985). Factoring Sparse Multivariate Polynomials. *J. of Computer and System Science* **31**, 265–287.

Grigoriev, D. Yu. and Karpinski, M. (1987). The Matching Problem for Bipartite Graphs with Polynomially Bounded Permanents is in NC. *Proceedings of the 28th IEEE Symposium on the Foundations of Computer Science*, 166–172.

Heintz, J., Sieveking, M. (1981). Absolute Primality of Polynomials is Decidable in Random Polynomial Time in the Number of Variables. *Lecture Notes in Computer Science, vol. 115*, 16–28.

Kaltofen, E. (1985a). Polynomial-Time Reductions from Multivariate to Bi- and Univariate Integral Polynomial Factorizations. *SIAM J. of Computing* **14**, 469–489.

Kaltofen, E. (1985b). Computing with Polynomials given by Straight-Line Programs I; Greatest common divisors. *Proceedings, 17th ACM Symposium on Theory of Computation*, 131–142.

Kaltofen, E. (1987). Factorization of Polynomials Given by Straight Line Programs..

Kaltofen, E., Yagati, L. (1988). *Improved Sparse Multivariate Polynomial Interpolation Algorithms*, Report 88–17, Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY.

Moses, J., Yun, D. Y. Y. (1973). The EZGCD Algorithm. *Proceedings of ACM National Conference*, 159–166.

Pólya, G., Szegö, G. (1976). *Problems and Theorems in Analysis*, Springer-Verlag, New York.

Press, W., Flannery, B. P., Teukolsky, S. A., Vetterling, W. T. (1986). *Numerical Recipes, The Art of Scientific Computing*, Cambridge University Press, Cambridge.

Rosser, J. B., Schoenfeld, L. (1962). Approximate Formulas for Some Functions of Prime Numbers. *Illinois Journal of Mathematics* **6**, 64–94.

Schwartz, J. T. (1980). Probabilistic Algorithms for Verification of Polynomial Identities. *J. Assoc. Comp. Mach.* **27**, 701–717.

Tiwari, P. (1987). *Parallel Algorithms for Instance of the Linear Matroid Parity with a Small Number of Solutions*, IBM Research Report RC 12766, IBM T. J. Watson Research Center, Yorktown Heights, NY.

van der Waerden, B. L. (1953). *Modern Algebra*, F. Ungar Publ. Co., New York.

Wang, P. S.-H., and Rothschild, L. P. (1975). Factoring Multivariate Polynomials over the Integers. *Math. Comp.* **29**, 935–950.

Wang, P. S.-H. (1978). An Improved Multivariate Polynomial Factoring Algorithm. *Math. Comp.* **32**, 1215–1231.

Zippel, R. E. (1979). Probabilistic Algorithms for Sparse Polynomials. *Lecture notes in Computer Science 72: Symbolic and Algebraic Computation*, 216-226.

Zippel, R. E. (1980). Newton's Iteration and the Sparse Hensel Algorithm. *Proceedings of SYMSAC'80*, 68–72.