FAST PARALLEL ALGORITHMS FOR SPARSE MULTIVARIATE POLYNOMIAL INTERPOLATION OVER FINITE FIELDS

DIMA YU. GRIGORIEV
STEKLOV INSTITUTE OF MATHEMATICS
SOV. ACAD. OF SCIENCES
LENINGRAD 191011

MAREK KARPINSKI*
DEPT. OF COMPUTER SCIENCE
UNIVERSITY OF BONN
5300 BONN 1

MICHAEL F. SINGER
DEPT. OF MATHEMATICS
N.C. STATE UNIVERSITY
RALEIGH, NC 27695

Abstract.

We design fast parallel deterministic algorithms (deterministic boolean NC) for interpolating arbitrary t-sparse polynomials over an *arbitrary* finite field $GF[q][x_1, \cdots, x_n]$. The Interpolation Algorithm works in a *slight* field extension $GF[q^{\lceil 2\log_q(nt)\rceil+3}]$ and uses $O(\log^3(nt))$ boolean parallel time and $O(n^2t^6\log^2 nt)$ processors. It is the first efficient deterministic polynomial time algorithm (and moreover boolean NC-algorithm) for this problem, and together with the efficient deterministic interpolation algorithms for fields of characteristic 0 (cf.[GK 87], [BT 88]) yields for the first time the general deterministic sparse conversion algorithm working over arbitrary fields. (The reason for this is that every field of positive characteristic contains a primitive subfield of this characteristics, and so we can apply our method to the slight extension of this subfield). The method of solution involves the polynomial enumeration techniques of [GK 87] combined with introducing a new general method of breaking the problem of identity to zero of sparse polynomials over the (logarithmic) *slight* field extension.

*Supported in part by Leibniz Center for Research in Computer Science and the DFG Grant KA 673/2-1

1. Introduction.

The sparse interpolation algorithms over finite fields play a key role in the design of efficient algorithms in algebra and their applications (cf. [G 83], [G 84], [K 85]; for applications in coding theory, see [LN 86], [MS 72]). These algorithms constituted a bottleneck in the design of efficient algorithms, and in particular no deterministic polynomial time algorithms were known for the sparse polynomial interpolation problems. The existing methods required large extension fields of order $GF[q^n]$, so, for example, no effective procedures even for finding primitive elements over an actual interpolation field were known without using randomization.

Here we remedy the situation by considering what we call a 'slight' extension of fields, which is an extension logarithmic in nt, $GF[q^{\lceil c\log_q(nt)\rceil}]$. The method of solution involves two major steps: (1) breaking the zero identity problem of polynomials from $GF[q]$ over a slight extension $GF[q^{\lceil 2\log_q(nt)+3\rceil}]$, and (2) inductive enumeration of partial solutions for terms and coefficients over $GF[q]$ by means of recursion on (1). We develop a general method involving Cauchy matrices to break the zero-identity in step 1, and combine this with the refined polynomial enumeration techniques of [GK 87] to solve step 2.

Because of the lower bound of $\Omega(n^{\log t})$ (cf. [CDGK 88]) for the interpolation over the same field $GF[q]$ without an extension, our *slight* field extension is in a sense the smallest extension possible to carry out the efficient interpolation.

In what follows we shall use the basic notions of the theory of finite fields (cf.[LN86], [MS 71]) and algorithms for computing in finite fields (cf. [L 82]), and the basic models of parallel computation (cf. [C 85], [G 82]).

2. Interpolation Problem over Finite Fields.

We consider the most general Problem of Interpolation for manipulating multivariate polynomials given by black boxes (a very special case of it are the interpolations of polynomials given by straight-line programs (cf.[K 85]), or polynomials given by determinants (cf. [L 79], [GK 87]). In this setting we are given a polynomial f in $GF[q]$ as a black box and information about its sparsity t (the bound on the number of its nonzero coefficients). Given this, we must determine an extension $GF[q^s]$ of $GF[q]$, s as small as possible, and an efficient polynomial time interpolation algorithm working over $GF[q^s]$ to determine all coefficients of f in $GF[q]$.

We say that the Interpolation Problem over a finite field $GF[q]$ is in NC^k (cf. [C 85]), if there exists a class of uniform $n^{O(1)}$-size and $O(\log^k n)$-depth boolean circuits with oracle nodes S (*returning* values of a black box over the field extension $GF[q^S]$) computing for an arbitrary n-variate polynomial $f \in GF[q][x_1, \cdots, x_n]$ all the nonzero coefficients and monomial vectors of f, with the oracle S_f^S, defined by $S_f^S(x_1, \cdots, x_n, y)$ iff $f(x_1, \cdots, x_n) = y$ over $GF[q^S]$. If the computation of the extension field $GF[q^S]$ and the computation of $f(x_1, \cdots, x_n)$ over $GF[q^S]$ performed by a black box (straight-line program, determinant, boolean circuit, etc.) are both in NC (in P), then the Interpolation Problem is also in NC (in P).

We note that the Interpolation Problem over finite fields deals not only with the interpolation of polynomials but arbitrary functions in their t-sparse ring sum expansion representation (RSE) ([W 86]).

We shall develop an interpolation algorithm (for polynomials over $GF[q]$) for the *slight* extension of a field of order $s = \lceil 2\log(nt) + 3 \rceil$. This allows us for the first time to find the generators in $GF[q^s]$, as the size of this field is polynomial in the size of the input polynomial under interpolation. Our *slight* field extension is in a sense the best possible, as the efficient

interpolation over the same field (i.e. for $s = 1$) is not possible. In [CDGK 88] the tight lower and upper bounds $\Theta(n^{\log t})$ have been established for the number of steps needed to determine identity to zero of polynomials $f \in GF[2][x_1, \cdots, x_n]$.

3. The Algorithm.

We formulate now the Interpolation Theorem and the underlying Interpolation Algorithm over Finite Fields.

Main Theorem. Given any t-sparse polynomial $f \in GF[q][x_1, \cdots, x_n]$. For an arbitrary q, there exists a deterministic parallel algorithm (NC^3) for interpolating f over a *slight* field extension $GF[q^{\lceil 2\log_q(nt)+3 \rceil}]$ working in $O(\log^3(ntq))$ parallel boolean time and $O(n^2t^6\log^2(ntq) + q^{2.5}\log^2 q)$ processors. For a fixed field the algorithm works in $O(\log^3(nt))$ parallel boolean time and $O(n^2t^6\log^2 nt)$ processors.

Sparse Interpolation Algorithm over Finite Fields.

Input: Black-box oracle allowing one to evaluate a t-sparse polynomial $f \in GF[q^s][x_1, \cdots, x_n]$ for $s = 1, \ldots$. (A t-sparse polynomial is a polynomial with at most t nonzero coefficients.)

Output: All $(\mathbf{k}, f_{\mathbf{k}})$ such that $f = \sum f_{\mathbf{k}} x^{\mathbf{k}}$ where $0 \neq f_{\mathbf{k}} \in GF[q]$ and $x^{\mathbf{k}} = x_1^{k_1} \cdot \ldots \cdot x_n^{k_n}$

We begin by first describing a **Subalgorithm.**

Subalgorithm (Identity-To-Zero Test):

Input: Same as above.

Output: Yes, if $f \equiv 0$; No, if $f \not\equiv 0$.

Step 1: Choose s so that $q^s - 1 > 4nq(n-1)\binom{t}{2}$. So let $s = \lceil 2\log_q(nt) + 3 \rceil$.

Step 2: Construct the field $GF[q^s]$ by looking over all polynomials of degree s with coefficients in $GF[q]$ and testing irreducibility with the help of the Berlekamp Algorithm [B 10]. We find an irreducible $\phi \in GF[q][z]$, and then $GF[q^s]$ is isomorphic to $GF[q][z]/(\phi)$. We find an ω that is a generator of the cyclic group $GF[q^s]^*$ in the following way. Factor $q^s - 1 = \prod p_i^{n_i}$, p_i prime. For any $a \in GF[q^s]$, calculate $a^{\frac{q^s-1}{p_i}}$ for each i. We do this using the binary expansion of the exponent and by techniques from [L 82]. An element is a generator of the cyclic group if and only if all these powers are distinct from 1.

Step 3: Denote $N = \frac{\lceil q^s - 1 \rceil}{4nq}$. Use the sieve of Eratosthenes to find a prime p with $2N < p \leq 4N$.

Step 4: Construct now an $N \times N$ Cauchy matrix C (cf. [C], [PS 64], [MS 71]) over the field $GF[p], y_i = x_i = i, 1 \leq i \leq N$ by $C = \left[\frac{1}{x_i + y_j}\right] = \left[\frac{1}{i+j}\right]$. We have

$$det\, C = \frac{\prod_{1 \leq i < j \leq n}(x_j - x_i)(y_j - y_i)}{\prod_{1 \leq i,j \leq n}(x_i + y_j)}.$$

For any of its minors $\neq 0$, a similar formula holds. Therefore any minor of any size is nonsingular.

Step 5: Compute, using the Euclidean algorithm $c_{ij} \in \mathbb{Z}$, such that $c_{ij} \equiv 1/_{i+j} (mod\, p)$ and $0 \leq c_{ij} < p \leq 4N$.

Step 6: Denote by $\overline{C} = [\overline{c_{ij}}]$ an arbitrary submatrix of C of size $N \times n$.

Step 7: Pick out in parallel any row $\overline{c_i} = (\overline{c_{ij}}), 1 \leq j \leq n$, of the matrix $\overline{C}$ and, for each $\ell, 0 \leq \ell < t$, plug $\omega^{\ell \overline{c_{ij}}}$ for each x_j in the black-box (with $s = \lceil 2\log_q(nt) + 3 \rceil$) for the

polynomial $f = \sum f_{\boldsymbol{k}} x^{\boldsymbol{k}} = \sum f_{\boldsymbol{k}} x_1^{k_1} \cdots x_n^{k_n}$, where $\boldsymbol{k} = (k_1, \cdots, k_n)$ and the number of $\boldsymbol{k}$'s is less than t, $0 \leq k_j < q-1$, $f_{\boldsymbol{k}} \in GF[q]$. □

We now pause to justify that if $f \not\equiv 0$ then for some $\overline{c_{ij}}\ell$ as above $f(\omega^{\ell \overline{c_i}}) \neq 0$, where $\omega^{\ell \overline{c_{ij}}}$ has been substituted for x_j. We first show that for a suitable vector $\overline{c_i}$, $1 \leq i \leq N$, after substituting $\omega^{\overline{c_{ij}}}$ for x_j, any two monomials $x^{\boldsymbol{k}}$, $x^{\boldsymbol{k}'}$ would give different elements of $GF[q]$. Suppose that for some pair $\boldsymbol{k}$, $\boldsymbol{k}'$ and $\overline{c_i}$ we have $\omega^{\overline{c_i} \cdot \boldsymbol{k}} = \omega^{\overline{c_i} \cdot \boldsymbol{k}'}$. This means that $\sum k_j \overline{c_{ij}} \equiv \sum k'_j \overline{c_{ij}} (mod\, q^s - 1)$ and so $\sum (k_j - k'_j) \overline{c_{ij}} \equiv 0 (mod\, q^s - 1)$. Since $|k_j - k'_j| \leq q-1, \overline{c_{ij}} < 4N$, we have $|\sum_{1 \leq j \leq n} (k_j - k'_j) \overline{c_{ij}}| < (q-1) n 4N < (q^s - 1)$, therefore $\sum (k_j - k'_j) c_{ij} = 0$. For any pair of monomials $x^{\boldsymbol{k}}, x^{\boldsymbol{k}'}$, we consider all the "bad" vectors $\overline{c_i}, 1 \leq i \leq N$, i.e those $\overline{c_i}$ for which $\sum_{1 \leq j \leq n} (\boldsymbol{k}_j - \boldsymbol{k}') \overline{c_{ij}} = 0$. There cannot be more than $(n-1)$ "bad" vectors for this pair, since if there exist such n vectors $\overline{c_{i_1}}, \cdots, \overline{c_{i_n}}$, the corresponding $n \times n$ submatrix of $\overline{C}$ would have determinant zero. As there are at most $\binom{t}{2}$ pairs of monomials, there is a vector $\overline{c_{i_0}}$, $1 \leq i_0 \leq N$ which is not "bad" for any pair of monomials $\boldsymbol{k}, \boldsymbol{k}'$ since $\binom{t}{2}(n-1) < N$.

Let $\overline{c_{i_0}}$ be some vector such that distinct monomials $x^{\boldsymbol{k}}, x^{\boldsymbol{k}'}$ yield distinct elements of $GF[q^s]$ after substituting $\omega^{\overline{c_{i_0}}}$. We now show that $f(\omega^{\ell \overline{c_{i_0}}}) \neq 0$ for some $0 \leq \ell < t$. If $f(\omega^{\ell \overline{c_{i_0}}}) = 0$ for all $\ell, 0 \leq \ell < t$, then $XV = 0$, where $X = (f_{\boldsymbol{k}})_{\boldsymbol{k}}$ and $V = (\omega^{\ell \overline{c_{i_0}} \cdot \boldsymbol{k}})$ is the $t \times t$ matrix whose rows are indexed by ℓ, $0 \leq \ell < t$ and columns are indexed by the $\boldsymbol{k}$ that appear as exponents in f.

Note that $det(V)^2 = \prod_{\boldsymbol{k} \neq \boldsymbol{k}'} (\omega^{\sum_j \overline{c}_{i_0 j} k_j} - \omega^{\sum_j \overline{c}_{i_0 j} k'_j}) \neq 0$ (it is a Vandermonde Matrix), so we have a contradiction. Therefore the identity-to-zero subalgorithm is correct.

We now continue with the main algorithm. Assume $n = 2^m$ for simplicity of notation. Define

$S_{\alpha,\beta} = \{(k_1, \cdots, k_{2^{\alpha-1}}) : x_{\beta 2^{\alpha-1}+1}^{k_1} \cdots x_{\beta 2^{\alpha-1}+2^{\alpha-1}}^{k_{2^{\alpha-1}}}$ occurs as a subterm in some nonzero term of $f\}$, where $1 \leq \alpha \leq m+1$ and $0 \leq \beta < 2^{m+1-\alpha}$. We produce $S_{\alpha,\beta}$ recursively for $\alpha = 1, \cdots, m+1$.

Basis Step: Let $\alpha = 1$ .In parallel for each $a \in GF[q]$, substitute a for $x_{\beta+1}$ in f. Find a vector $u_\ell \in (GF[q])^q$ such that $u_\ell \cdot (a_i^j) = (0, \cdots, 1, \cdots, 0)$ where all entries of this latter vector are 0 except for a 1 in the ℓ^{th} place. We then have $u_\ell \cdot (f(x_1, \cdots, x_\beta, a_1, x_{\beta+2}, \cdots, x_n), \cdots, f(x_1, \cdots, x_\beta, a_q, x_{\beta+2}, \cdots, x_n)) = P_\ell$ where $f = \sum_\ell x_{\beta+1}^\ell P_\ell$ and $P_\ell \in GF[q][x_1, \cdots, x_\beta, x_{\beta+2}, \cdots, x_n]$. Using this last formula with black box evaluations of f, gives us a new black box for each P_ℓ. The identity-to-zero subalgorithm now allows us to determine which P_ℓ's are not identically zero, and so to determine $S_{1,\beta}$.

Recursion Step: Assume that we have produced $S_{\alpha,\beta}$ for all $\beta, 0 \leq \beta < 2^{m+1-\alpha}$. We now produce $S_{\alpha+1,\beta}$ for fixed β, $0 \leq \beta < 2^{m-\alpha}$. For each element from the set $S_{\alpha,2\beta}$ and for each element from the set $S_{\alpha,2\beta+1}$, consider the corresponding product $x_{\beta 2^{\alpha}+1}^{k} \cdots x_{\beta 2^{\alpha}+2^{\alpha}}^{k_{2^{\alpha}}}$. For all such products (observe that the number of them is at most t^2 since $|S_{\alpha,2\beta}|, |S_{\alpha,2\beta+1}| \leq t$), we can find (in parallel) a vector $v \in I\!N^{2^\alpha}$ as in step 7 such that $v = (v_1, \cdots, v_{2^\alpha})$, $0 \leq v_i < 4N_1$, where s_1 is chosen such that $\frac{\lceil q^{s_1}-1 \rceil}{4nq} = N_1 > (n-1)\binom{t^2}{2}$ and for any two products $x_{\beta 2^\alpha+1}^{k_1} \cdot \cdots \cdot x_{\beta 2^\alpha+2^\alpha}^{k_{2^\alpha}}$ and $x_{\beta 2^\alpha+1}^{k'_1} \cdot \cdots \cdot x_{\beta 2^\alpha+2^\alpha}^{k'_{2^\alpha}}$, $q^{s_1}-1 \nmid (\sum k_i v_i - \sum k'_i v_i)$. Let $\omega_1 \in GF[q^{s_1}]$ be a generator of the cyclic group $GF[q^{s_1}]^*$. For any $0 \leq l < t^2$, we replace $x_{\beta 2^\alpha+j}$ with $\omega_1^{v_j \ell}$. Consider the $t^2 \times t^2$ matrix $B = (\omega_1^{(\sum_j k_j v_j)\ell}) = (b_{\boldsymbol{k}}, \ell)$. Note that $det(B)^2 = \prod_{\boldsymbol{k} \neq \boldsymbol{k}'} (\omega_1^{(\sum_j k_j v_j)} - \omega_1^{(\sum_j k'_j v_j)}) \neq 0$ since $q^{s_1} - 1 \nmid (\sum_j k_j v_j - \sum k'_j v_j)$. Calculate vectors $u_j \in (GF[q^{s_1}])^{t^2}$ such that $u_j B = (0, \cdots, 0, 1, 0, \cdots, 0)$ where this latter vector has 1 in the i-th position and zeroes everywhere else. We then have $u_i \cdot Y = P_i$ where $f = \sum_{\boldsymbol{k}} x^{\boldsymbol{k}} \bar{P}_{\boldsymbol{k}}$ where $x^{\boldsymbol{k}} = x_{\beta 2^\alpha+1}^{k_1} \cdot \cdots \cdot x_{\beta 2^\alpha+2^\alpha}^{k_{2^\alpha}}$ and

$\bar{P}_{\boldsymbol{k}} \in GF[q][x_1, \cdots, x_{\beta 2^\alpha}, x_{(\beta+1)2^\alpha+1}, \cdots, x_n]$ and Y is the $1 \times t^2$ vector whose ℓ^{th} entry is $f(x_1, \cdots, x_{\beta 2^\alpha}, \omega_1^{v_1 \ell}, \cdots, \omega_1^{v_{2^\alpha} \ell}, x_{(\beta+1)2^\alpha+1}, \cdots, x_n)$. Using this last formula with black box evaluations of f gives us the new black boxes for the $\overline{P_i}$. The **identity-to-zero** subalgorithm now allows us to determine which $\overline{P_i}$ are not identically zero and so to determine $S_{\alpha+1,\beta}$. Notice that when $\alpha = m+1$ we have determined all the terms of f in the form of $(\boldsymbol{k}, f_{\boldsymbol{k}})$ such that $f = \sum_{\boldsymbol{k}} f_{\boldsymbol{k}} x^{\boldsymbol{k}}, 0 \neq f_{\boldsymbol{k}} \in F[q]$ and $x^{\boldsymbol{k}} = x_1^{k_1}, \cdots, x_n^{k_n}$. □

4. Analysis of the Algorithm.

Let $N = \frac{\lceil q^s - 1 \rceil}{4nq}$. Note that $N < nt^2q$. The parallel time of our algorithm is $O(\log^3 N)$. This is because the **identity-to-zero** test takes $O(\log^2 N)$ parallel time, the recursive step calls this test and uses matrix inversion, which requires $O(\log^2 N)$ parallel time [M 86], and the recursion depth is $O(\log n)$. Steps 1-6 take $O(N \log^2(Nnq))$ processors. Step 7 takes $O(Nnt \log^2(Nnq))$ processors. Therefore the total cost (in processors) of the identity-to-zero subalgorithm is $O(Nnt \log^2(Nnq))$.

We now proceed to analyse the complexity of the rest of the algorithm.In the basic step, we must invert the $q \times q$ matrix (a_i^j) over $GF[q]$. This requires $O(q^{2.5} \log^2 q)$ processors by [M 86]. In applying steps 1-7 to test whether P_ℓ is identically zero, we refer q times to substituting $\omega^{\xi_{ij}}$ in a black box and calling the **identity-to-zero** test. Thus we need $Nntq \log^2 Nnq$ processors. In the recursion step, we calculate $N_1 t^2$ sums $\sum_j k_j v_j$ of length n and compute $\omega_1^{\sum_j k_j v_j}$ in the field $GF[q^{S_1}]$. This takes $N_1 t^2 n \log^2 N_1$ processors. Notice that $N_1 < nt^4 q$. Inverting the $t^2 \times t^2$ matrix B over $GF[q^{S_1}]$ requires $t^5 \log^2 N_1$ processors [M 86]. Therefore the total number of processors would be $O(t^6 n^2 q \log^2(tnq) + q^{2.5} \log^2 q)$. For a fixed field, the algorithm

works in $O(\log^3 nt)$ time and $O(n^2t^6 \log^2 nt)$ processors.

5. Further Research.

Our parallel algorithm enjoys very good parallel time bound. Concerning the number of processors, would it be possible to improve on the number of processors of the Interpolation Algorithm?

Acknowledgements.

We are grateful to Michael Ben-Or, Johannes Grabmeier, Michael Rabin, Volker Strassen, and Avi Wigderson for a number of interesting conversations.

References.

[AL 86] Adleman, L.M., and Lenstra, H.K., *Finding Irreducible Polynomials over Finite Fields*, Proc. ACM STOC (1986), pp. 350-355

[B 70] Berlekamp, E.R., *Factoring Polynomails over Large Finite Fields*, Math. Comp. 24 (1970), pp. 713-735

[B 81] Ben-Or, M., *Probabilistic Algorithms in Finite Fields*, Proc. 22^{nd} IEEE FOCS (1981), pp. 394-398

[BT 88] Ben-Or, M., and Tiwari, P., *A deterministic Algorithm for Sparse Multivariate Polynomail Interpolation*, Proc. 20^{th} ACM STOC (1988)

[C] Cauchy, A.L., *Exercises d'Analyse et de Phys. Math., Vol 2*, Paris, Bachelier (1841), pp. 151-159

[C 85] Cook, S.A., *A taxonomy of Problems with Fast Parallel Algorithms*, Information and Control 64 (1985), pp. 2-22

[CDGK 88] Clausen, M., Dress, A., Grabmeier, J., and Karpinski, M., *On Zero-Testing and Interpolation of k-Sparse Multivariate Polynomials over Finite Fields*, Research Report No. 8522-CS, University of Bonn, pp. 1-16 (May 1988)

[G 82] Goldschlager, L., *Synchronous Parallel Computation*, J. ACM 29 (1982), pp. 1073-1086

[G 83] von zur Gathen, J., *Factoring Sparse Multivariate Polynomials*, Proc. 24^{th} IEEE FOCS (1983), pp. 172-179

[G 84] von zur Gathen, J., *Parallel Algorithms for Algebraic Problems*, SIAM J. Comput., Vol. 13 (1984), pp. 808-824

[GK 87] Grigoriev, D.Yu, and Karpinski, M., *The Matching Problem for Bipartite Graphs with Polynomially Bounded Permanents Is in NC*, Proc. 28^{th} IEEE FOCS (1987), pp. 166-172

[K 85] Kaltofen, E., *Computing with Polynomials Given by Straight-Line Programs In Greatest Common Divisors*, Proc. 17^{th} ACM STOC (1985), 131-142

[L 79] Lovasz, L., *On Determinants, Matchings, and Random Algorithms*, in: Fundamentals of Computation Theory, Akademie-Verlag, Berlin 1979, pp. 565-574

[L 82] Loos, R., *Computing in Algebraic Extensions*, in: Computer Algebra: Symbolic and Algebraic Computation, Springer-Verlag 1982, pp. 173-187

[L 83] Lenstra, A.K., *Factoring Multivariate Polynomials over Finite Fields*, Proc. 15^{th} ACM STOC (1983), pp. 189-192

[LN 86] Lidl, H., and Niederreiter, H., *Introduction to Finite Fields and Their Applications*, Cambridge University Press 1986

[MS 72] MacWilliams, F.J., and Sloane, N.J.A., *The Theory of Error-Correcting Codes*, North Holland, Amsterdam 1977

[M 86] Mulmuley, K., *A fast Parallel Algorithm to Compute the Rank of a Matrix over an Arbitrary Field*, Proc. 18^{th} ACM STOC (1986), pp. 338-339

[PS 64] Polya, G., and Szegö, G., *Aufgaben und Lehrsätze aus der Analysis*, Vol. 2, Springer-Verlag, Berlin 1964

[S 80] Schwartz, J.T., *Fast Probabilistic Algorithms for Verification of Polynomial Identities*, J. ACM 27,4 (1980), pp. 701-717

[W 87] Wegener, I., *The Complexity of Boolean Functions*, Wiley 1987

[Z 79] Zippel, R.E., *Probabilistic Algorithms for Sparse Polynomials*, Proc. EUROSAM '79, Springer Lecture Notes in Computer Science Vol 72 (1979), pp. 216-226