SIAM J. COMPUT.
Vol. 19, No. 6, pp. 1059-1063, December 1990

007

FAST PARALLEL ALGORITHMS FOR SPARSE MULTIVARIATE POLYNOMIAL INTERPOLATION OVER FINITE FIELDS*

DIMA YU. GRIGORIEV†, MAREK KARPINSKI‡, AND MICHAEL F. SINGER§

Abstract. The authors consider the problem of reconstructing (i.e., interpolating) a t-sparse multivariate polynomial given a *black box* which will produce the value of the polynomial for any value of the arguments. It is shown that, if the polynomial has coefficients in a finite field $GF[q]$ and the black box can evaluate the polynomial in the field $GF[q^{\lceil 2\log_q(nt)+3\rceil}]$, where n is the number of variables, then there is an algorithm to interpolate the polynomial in $O(\log^3(nt))$ boolean parallel time and $O(n^2t^6\log^2 nt)$ processors.

This algorithm yields the first efficient deterministic polynomial time algorithm (and moreover boolean NC-algorithm) for interpolating t-sparse polynomials over finite fields and should be contrasted with the fact that efficient interpolation using a black box that only evaluates the polynomial at points in $GF[q]$ is not possible (cf. [M. Clausen, A. Dress, J. Grabmeier, and M. Karpinski, *Theoret. Comput. Sci.*, 1990, to appear]). This algorithm, together with the efficient deterministic interpolation algorithms for fields of characteristic 0 (cf. [D. Yu. Grigoriev and M. Karpinski, in *Proceedings of the 28th IEEE Symposium on the Foundations of Computer Science*, 1987, pp. 166-172], [M. Ben-Or and P. Tiwari, in *Proceedings of the 20th ACM Symposium on the Theory of Computing*, 1988, pp. 301-309]), 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 characteristic, and so this method can be applied to the slight extension of this subfield.) The method of solution involves the polynomial enumeration techniques of [D. Yu. Grigoriev and M. Karpinski, *op. cit.*] combined with introducing a new general method of solving the problem of determining if a t-sparse polynomial is identical to zero by evaluating it in a *slight* extension of the coefficient field (i.e., an extension whose degree over this field is logarithmic in nt).

Key words. sparse multivariate polynomials, finite fields, interpolation

AMS(MOS) subject classifications. 68C25, 12C05

1. Introduction. The polynomial interpolation algorithms play an important role in the design of efficient algorithms in algebra and their applications (cf. [G83], [G84], [K85], [BT88]). For the case of finite fields there were no deterministic polynomial time algorithms known (cf. [BT88]) for the sparse interpolation problem. The existing methods required large extension fields of order $GF[q^n]$; so, for example, no effective procedures 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 whose degree over the coefficient field is logarithmic in nt, $GF[q^{\lceil c\log_q(nt)\rceil}]$. The method of solution involves two major steps: (1) solving the zero identity problem of polynomials from $GF[q]$ by evaluating in a slight extension $GF[q^{\lceil 2\log_q(nt)+3\rceil}]$, and (2) using 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 solve the zero-identity problem in Step 1, and combine this with the refined polynomial enumeration techniques of Grigoriev and Karpinski [GK87] to solve Step 2.

Because of the lower bound of $\Omega(n^{\log t})$ (cf. [CDGK88]) for the interpolation over the same field $GF[q]$ without an extension, our *slight* field extension is in a sense the smallest extension capable of carrying out the efficient interpolation.

* Received by the editors March 17, 1989; accepted for publication (in revised form) March 4, 1990.
† Steklov Institute of Mathematics, Soviet Academy of Sciences, Leningrad 191011, USSR.
‡ Department of Computer Science, University of Bonn, 5300 Bonn 1, Federal Republic of Germany.
The work of this author was supported in part by the Leibniz Center for Research in Computer Science and the Deutsche Forschungsgemeinschaft grant KA673/2-1.
§ Department of Mathematics, North Carolina State University, Raleigh, North Carolina 27695.

In what follows we shall use the basic notions of the theory of finite fields (cf. [LN86], [MS77]) and algorithms for computing in finite fields (cf. [L82]), and the basic models of parallel computation (cf. [C85], [G82]).

2. Interpolation problem over finite fields. We consider the problem of interpolation for multivariate polynomials given by *black boxes* (special cases of it are the explicit interpolations of polynomials given by straight-line programs (cf. [K85]), or polynomials given by determinants (cf. [L79], [GK87]). In this setting we are given a polynomial f in $GF[q]$ as a black box that allows us to evaluate f in extensions of $GF[q]$ 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 *black box* interpolation problem (over a finite field extension $GF[q^S]$) is in NC^k (cf. [C85]), if there exists a class of uniform $(ntq)^{0(1)}$-size and $O(\log^k(ntq))$-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 $S_f^S(x_1,\cdots,x_n,y)$ if and only if $f(x_1,\cdots,x_n)=y$ over $GF[q^S]$. If the *lifting* of a *black box* (given explicitly by a straight-line program, determinant, boolean circuit, etc.) from $GF[q]$ to the extension $GF[q^S]$, and the computation of $f(x_1,\cdots,x_n)$ over $GF[q^S]$ by a black box, are both in boolean NC (in P), then the explicit interpolation problem lies also in boolean NC (in P).

We note that the interpolation problem over finite fields deals not only with the interpolation of polynomials but with arbitrary functions in their t-sparse ring sum expansion representation (RSE) ([W87]).

We shall develop an interpolation algorithm (for polynomials over $GF[q]$) for the *slight* extension of a field of order $s=\ulcorner 2\log(nt)+3\urcorner$. This allows us for the first time to efficiently 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 [CDGK88] 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 now formulate the Interpolation Theorem and the underlying Interpolation Algorithm over Finite Fields.

THE INTERPOLATION 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 $\underline{\text{slight}}$ field extension $GF[q^{\ulcorner 2\log_q(nt)+3\urcorner}]$ 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: A black-box oracle allowing one to evaluate a t-sparse polynomial $f\in GF[q^s][x_1,\cdots,x_n]$ for $s=1,\cdots$. (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 $s = \ulcorner 2 \log_q (nt) + 3 \urcorner$.

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 [B70]. 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 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 $a^{(q^s-1)/p_i}$ for each i. We do this using the binary expansion of the exponent and by techniques from [L82]. An element is a generator of the cyclic group if and only if all these powers are distinct from 1.

Step 3: Denote $N = (\ulcorner q^s - 1 \urcorner)/4nq$. Use the sieve of Eratosthenes to find a prime p with $2N < p \leqq 4N$. Such a prime exists by Bertrand's postulate (cf. [HW78]).

Step 4: Now construct an $N \times N$ Cauchy matrix C (cf. [C], [PS64], [MS77]) over the field $GF[p]$, $y_i = x_i = i$, $1 \leqq i \leqq N$ by $C = [1/(x_i + y_j)] = [1/(i+j)]$. We have

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

For any of its minors $\neq 0$, a similar formula holds. Therefore any minor of any size is nonsingular. Compute, using the Euclidean algorithm $c_{ij} \in \mathbb{Z}$, such that $c_{ij} \equiv 1/(i+j) (\text{mod } p)$ and $0 \leqq c_{ij} < p \leqq 4N$.

Step 5: Denote by $\bar{C} = [\bar{c}_{ij}]$ an arbitrary submatrix of C of size $N \times n$.

Step 6: Pick out in parallel any row $\bar{c}_i = (\bar{c}_{ij})$, $1 \leqq j \leqq n$, of the matrix $\bar{C}$ and, for each l, $0 \leqq l < t$, plug $\omega l \bar{c}_{ij}$ for each x_j in the black-box (with $s = \ulcorner 2 \log_q (nt) + 3 \urcorner$) for the polynomial $f = \sum f_{\mathbf{k}} x^{\mathbf{k}} = \sum f_{\mathbf{k}} x_1^{k_1} \cdots x_n^{k_n}$, where $\mathbf{k} = (k_1, \cdots, k_n)$ and the number of $\mathbf{k}$'s is less than t, $0 \leqq k_j < q-1$, $f_{\mathbf{k}} \in GF[q]$.

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

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

Note that $\det (V)^2 = \prod_{\mathbf{k} \neq \mathbf{k}'} (\omega^{\sum \bar{c}_{i_0 j} k_j} - \omega^{\sum \bar{c}_{i_0 j} k_j'}) \neq 0$ (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 $S_{\alpha,\beta} = \{(k_1, \cdots, k_{2^{\alpha-1}})\colon x_{\beta 2^{\alpha-1}+1}^{k_1} \cdot \ldots \cdot x_{\beta 2^{\alpha-1}+2^{\alpha-1}}^{k_{2^{\alpha-1}}}$ occurs as a subterm in some nonzero term of $f\}$, where $1 \leqq \alpha \leqq m+1$ and $0 \leqq \beta < 2^{m+1-\alpha}$. We produce $S_{\alpha,\beta}$ recursively for $\alpha = 1, \cdots, m+1$.

Basis Step: Let $\alpha=1$. Let $\{a_1, a_2, \cdots\}$ be an enumeration of $GF[q]$. In parallel for each $a \in GF[q]$, substitute a for $x_{\beta+1}$ in f. Find a vector $u_l \in (GF[q])^q$ such that $u_l \cdot (a_i^j) = (0, \cdots, 1, \cdots, 0)$ where all entries of this latter vector are 0 except for a 1 in the lth place. We then have $u_l \cdot (f(x_1, \cdots, x_\beta, a_1, x_{\beta+2}, \cdots, x_n), \cdots, f(x_1, \cdots, x_\beta, a_q, x_{\beta+2}, \cdots, x_n)) = P_l$ where $f=\sum_l x_{\beta+1}^l P_l$ and $P_l \in GF[q][x_1, \cdots, x_\beta, x_{\beta+2}, \cdots, x_n]$. We see that P_l may be evaluated at any point $(b_1, \cdots, b_{\beta-1}, b_{\beta+1}, \cdots, b_n)$ by evaluating f at the q points $(b_1, \cdots, b_{\beta-1}, a_i, b_{\beta+1}, \cdots, b_n)$, $i = 1, \cdots, q$ and using this last formula, where u_l has been found by inverting the matrix (a_i^j) and extracting the lth row. This gives a black box for 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 β, $0 \leqq \beta < 2^{m+1-\alpha}$. We now produce $S_{\alpha+1,\beta}$ for fixed β, $0 \leqq \beta < 2^{m-\alpha}$. 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 $|S_{\alpha,2\beta}|, |S_{\alpha,2\beta+1}| \leqq t$), we can find (in parallel) a vector $v \in \mathbb{N}^{2^\alpha}$ as in Step 6 such that $v=(v_1, \cdots, v_{2^\alpha})$, $0 \leqq v_i < 4N_1$, where s_1 is chosen such that $(\ulcorner q^{s_1} - 1\urcorner)/4nq = N_1 > (n-1)\binom{t^2}{2}$ and for any two products $x_{\beta 2^\alpha+1}^{k_1} \cdot \ldots \cdot x_{\beta 2^\alpha+2^\alpha}^{k_{2\alpha}}$ and $x_{\beta 2^\alpha+1}^{k_1'} \cdot \ldots \cdot x_{\beta 2^\alpha+2^\alpha}^{k_{2\alpha}'}$, $q^{s_1}-1 \nmid (\sum k_i v_i - \sum k_1' v_i)$. Let $\omega_1 \in GF[q^{s_1}]$ be a generator of the cyclic group $GF[q^{s_1}]^*$. For any $0 \leqq l < t^2$, we replace $x_{\beta 2^\alpha+j}$ with $\omega_1^{v_j l}$. Consider the $t^2 \times t^2$ matrix $B=(\omega_1^{(\sum_j k_j v_j) l}) = (b_{\mathbf{k}}, l)$. Note that $\det(B)^2 = \prod_{\mathbf{k} \neq \mathbf{k}'} (\omega_1^{(\sum_j k_j v_j)} - \omega_1^{(\sum_j k_j' v_j)}) \neq 0$, 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 ith position and zeros everywhere else. We then have $u_i \cdot Y = \bar{P}_i$ where $f=\sum_{\mathbf{k}} x^{\mathbf{k}} \bar{P}_{\mathbf{k}}$, where $x^{\mathbf{k}} = x_{\beta 2^\alpha+1}^{k_1} \cdot \ldots \cdot x_{\beta 2^\alpha+2^\alpha}^{k_{2\alpha}}$ and $\bar{P}_{\mathbf{k}} \in GF[q][x_1, \cdots, x_{\beta 2^\alpha}, x_{(\beta+1)2^\alpha+1}, \cdots, x_n]$, and Y is the $1 \times t^2$ vector whose lth entry is $f(x_1, \cdots, x_{\beta 2^\alpha}, \omega_1^{v_1 l}, \cdots, \omega_1^{v_{2^\alpha} l}, x_{(\beta+1)2^\alpha+1}, \cdots, x_n)$. Using this last formula with black box evaluations of f gives us the new black boxes for the $\overline{P_i}$ as before. The *identity-to-zero* subalgorithm now allows us to determine which $\overline{P_i}$ are not identically zero and thus to determine $S_{\alpha+1,\beta}$. Notice that when $\alpha=m+1$ we have determined all the terms of f in the form of $(\mathbf{k}, f_{\mathbf{k}})$ such that $f=\sum_{\mathbf{k}} f_{\mathbf{k}} x^{\mathbf{k}}$, $0 \neq f_{\mathbf{k}} \in F[q]$ and $x^{\mathbf{k}} = x_1^{k_1}, \cdots, x_n^{k_n}$. □

4. Analysis of the Algorithm. Let $N = (\ulcorner q^{s-1}\urcorner/4nq)$. Note that $N < nt^2 q$. 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 [M86], and the recursion depth is $O(\log n)$. Steps 1-5 take $O(N \log^2(Nnq))$ processors. Step 6 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 analyze 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 [M86]. In applying Steps 1-6 to test whether P_l is identically zero, we refer q times to substituting $\omega^{\bar{c}_{ij}}$ 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 [M86]. 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 $O(n^2 t^6 \log^2 nt)$ 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?

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

REFERENCES

[AL86] L. M. ADLEMAN AND H. K. LENSTRA, *Finding irreducible polynomials over finite fields*, in Proceedings of the 18th ACM Symposium on the Theory of Computing, 1986, pp. 350-355.

[B70] E. R. BERLEKAMP, *Factoring polynomials over large finite fields*, Math. Comp., 24 (1970), pp. 713-735.

[B81] M. BEN-OR, *Probabilistic algorithms in finite fields*, in Proceedings of the 22nd IEEE Symposium on the Foundations of Computer Science, 1981, pp. 394-398.

[BT88] M. BEN-OR AND P. TIWARI, *A deterministic algorithm for sparse multivariate polynomial interpolation*, in Proceedings of the 20th ACM Symposium on the Theory of Computing, 1988, pp. 301-309.

[C] A. L. CAUCHY, *Exercises d'analyse et de physics mathematiques*, Vol. 2, Bachelier, Paris, 1841, pp. 151-159.

[C85] S. A. COOK, *A taxonomy of problems with fast parallel algorithms*, Inform. and Control, 64 (1985), pp. 2-22.

[CDGK] M. CLAUSEN, A. DRESS, J. GRABMEIER, AND M. KARPINSKI, *On zero-testing and interpolation of k-sparse multivariate polynomials over finite fields*, Theoret. Comput. Sci., 1990, to appear.

[G82] L. GOLDSCHLAGER, *Synchronous parallel computation*, J. Assoc. Comput. Mach., 29 (1982), pp. 1073-1086.

[G83] J. VON ZUR GATHEN, *Factoring sparse multivariate polynomials*, in Proceedings of the 24th IEEE Symposium on the Foundations of Computer Science, 1983, pp. 172-179.

[G84] ———, *Parallel algorithms for algebraic problems*, SIAM J. Comput., 13 (1984), pp. 808-824.

[GK87] D. YU. GRIGORIEV AND M. KARPINSKI, *The matching problem for bipartite graphs with polynomially bounded permanents is in* NC, in Proceedings of the 28th IEEE Symposium on the Foundations on Computer Science, 1987, pp. 166-172.

[HW78] G. H. HARDY AND E. M. WRIGHT, *An Introduction to the Theory of Numbers*, Fifth Edition, Oxford University Press, London, 1978.

[K85] E. KALTOFEN, *Computing with polynomials given by straight-line programs in greatest common divisors*, in Proceedings of the 17th ACM Symposium on the Theory of Computing, 1985, pp. 131-142.

[L79] L. LOVÀSZ, *On determinants, matchings, and random algorithms*, in Fundamentals of Computation Theory, Akademie-Verlag, Berlin, 1979, pp. 565-574.

[L82] R. LOOS, *Computing in algebraic extensions*, in Computer Algebra: Symbolic and Algebraic Computation, Springer-Verlag, Berlin, New York, 1982, pp. 173-187.

[L83] A. K. LENSTRA, *Factoring multivariate polynomials over finite fields*, in Proceedings of the 15th ACM Symposium on the Theory of Computing, 1983, pp. 189-192.

[LN86] H. LIDL AND H. NIEDERREITER, *Introduction to Finite Fields and Their Applications*, Cambridge University Press, London, 1986.

[MS72] F. J. MACWILLIAMS AND N. J. A. SLOANE, *The Theory of Error-Correcting Codes*, North-Holland, Amsterdam, 1977.

[M86] K. MULMULEY, *A fast parallel algorithm to compute the rank of a matrix over an arbitrary field*, in Proceedings of the 18th ACM Symposium on the Theory of Computing, 1986, pp. 338-339.

[PS64] G. PÒLYA AND G. SZEGÖ, *Aufgaben und Lehrsätze aus der Analysis*, Vol. 2, Springer-Verlag, Berlin, New York, 1964.

[S80] J. T. SCHWARTZ, *Fast probabilistic algorithms for verification of polynomial identities*, J. Assoc. Comput. Mach., 27 (1980), pp. 701-717.

[W87] L. WEGENER, *The Complexity of Boolean Functions*, John Wiley, New York, 1987.

[Z79] R. E. ZIPPEL, *Probabilistic algorithms for sparse polynomials*, in Proc. EUROSAM '79, Springer Lecture Notes in Computer Science, 72 (1979), pp. 216-226.