Reprinted from INFORMATION AND CONTROL
Vol. 52, No. 3, March 1982
All Rights Reserved by Academic Press, New York and London
Printed in Belgium

A. BORODIN, J. VON ZUR GATHEN & J. E. HOPCROFT (1982). Fast parallel matrix and GCD Computations. *Information and Control* **52**, 241–256. URL https://dx.doi.org/10.1016/S0019-9958(82)90766-5. Extended Abstract in *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, Chicago IL (1982).

This document is provided as a means to ensure timely dissemination of scholarly and technical work on a non-commercial basis. Copyright and all rights therein are maintained by the authors or by other copyright holders, notwithstanding that they have offered their works here electronically. It is understood that all persons copying any of these documents will adhere to the terms and constraints invoked by each copyright holder, and in particular use them only for noncommercial purposes. These works may not be posted elsewhere without the explicit written permission of the copyright holder. (Last update 2017/11/29-18:15.)

Fast Parallel Matrix and GCD Computations

ALLAN BORODIN,* JOACHIM VON ZUR GATHEN,* AND JOHN HOPCROFT†

*University of Toronto, Toronto, Canada, and
†Cornell University, Ithaca, New York

Parallel algorithms to compute the determinant and characteristic polynomial of matrices and the gcd of polynomials are presented. The rank of matrices and solutions of arbitrary systems of linear equations are computed by parallel Las Vegas algorithms. All algorithms work over arbitrary fields. They run in parallel time $O(\log^2 n)$ (where n is the number of inputs) and use a polynomial number of processors.

1. INTRODUCTION

Today's technology has motivated recent activity concerning parallel programs. Much of this activity has focussed on combinatorial questions (sorting, graph theoretic algorithms, etc.) and on questions relating to the parallel architecture itself (routing, queuing, etc.). It is also of recognized importance to investigate algebraic questions, and to this end we present algorithms for some basic problems such as computing the determinant and the rank of matrices or the gcd of polynomials.

There are two basically different approaches to what constitutes a "fast parallel algorithm." One is to start from a good sequential algorithm and try to parallelize it with a near-optimal speed-up, i.e., try to achieve parallel time close to (sequential time)/(number of processors). The second approach is to attempt to make the parallel time as small as possible, allowing an almost arbitrary (e.g., polynomially bounded) number of processors. While the first approach seems to be appropriate for the present technology, where in effect only a rather limited amount of parallelism is available, it is not unreasonable to expect that some time in the future the "asymptotically fast algorithms" of the second approach will play an important role. The situation is not unlike the dual approach to sequential algorithms, where one is interested in both constant speed-up of known algorithms (say, by

The work of the first two authors was partially supported by NSERC Grant A7631, and that of the third author by ONR Grant N00014-76-C-0018.

0019-9958/82 $2.00
Copyright © 1982 by Academic Press, Inc.
All rights of reproduction in any form reserved.

programming optimization) and the construction of asymptotically fast algorithms (even though the hidden constants for the computing time may be large). Perhaps the reader has guessed by now that here we pursue the second approach to parallel programming.

In this paper we discuss two basic problems: solving linear equations and simplifying rational expressions. Both have nice sequential solutions—Gaussian elimination and Euclid's algorithm—and it is an intriguing question if there also exist fast parallel methods. While Csanky (1976) has given a fast determinant algorithm over fields of characteristic zero, applications such as factoring polynomials require an algorithm that works over arbitrary fields, in particular finite fields. We present such an algorithm below, based on the general parallelization result by Valiant–Skyum–Berkowitz–Rackoff (1981).

As direct corollaries we get fast methods for inverting matrices and solving nonsingular systems of linear equations. Further applications are the characteristic polynomial of a matrix and the gcd of polynomials.

Some interesting combinatorial problems—maximal matchings, maximal flow—translate into the problem of computing the rank of matrices. We present a fast parallel Las Vegas method that either returns the rank of the input matrix or reports that it failed; the latter with small probability. Applications include finding a basis for the nullspace of a matrix, finding a maximal linearly independent subset of a given set of vectors, and the solution of a general (possibly singular) system of linear equations, all this again over arbitrary fields.

2. The Model

The algorithms described in this paper can be implemented on a synchronous shared-memory model of parallel computation such as the PRAM (Fortune–Wyllie, 1978), with arithmetic and tests in the ground field F as basic operations, or on an algebraic circuit. The algorithms all use $O(\log^2 n)$ parallel time (=depth for circuits) and $n^{O(1)}$ processors (=gates for circuits), when n is the number of inputs. In particular, it follows that the determinant and gcd problems are in the appropriate analog of uniform NC (Pippenger, 1979), and the rank problem is in the Monte Carlo or nonuniform analog of NC.

When the ground field F is $\mathbb{Q}$ or a finite field, we can represent the inputs as strings over a finite alphabet and ask for a (say) PRAM with bit instructions solving the problem. Our algorithms show that the determinant and gcd problems are in the corresponding Boolean class NC, and the rank problem is in the Monte Carlo or nonuniform Boolean version of NC (in fact, for $F = \mathbb{Q}$ in NC). Here the essential point is that (for $F = \mathbb{Q}$) according

to Edmonds (1967) the intermediate values in Gaussian elimination are reasonably small (see also Bareiss, 1968).

The basic stepping stone for the whole theory is the result by Valiant–Skyum–Berkowitz–Rackoff (1981). It says that any sequential program computing a polynomial of degree n with t steps can be converted to a parallel program with parallel time $O(\log n(\log n + \log t))$ using $O(t^3 n^6)$ processors.

3. Determinant and GCD

In this section we discuss the following problems: DETERMINANT(n) (=computing the determinant of an $n \times n$ matrix), CHARACTERISTIC POLYNOMIAL(n), NONSINGULAR EQUATIONS(n) (=computing the solution of a nonsingular $n \times n$ system of linear equations), INVERSION(n) (=computing the inverse of an $n \times n$ matrix, if it is nonsingular), GCD(n) (=computing the monic gcd(f, g), where $f, g \in F[x]$ have degree $\leqslant n$). We write, e.g., INVERSION for the sequence (INVERSION(n)).

The following result was proved by Csanky (1976), but only for fields of characteristic zero.

Theorem 1. *For any field* F, DETERMINANT, INVERSION, NONSINGULAR EQUATIONS, and CHARACTERISTIC POLYNOMIAL *can be computed in parallel time* $O(\log^2 n)$ *using a polynomially bounded number of processors. For* DETERMINANT *and* CHARACTERISTIC POLYNOMIAL, *neither branching nor division occurs.*

Proof. We consider ordinary Gaussian elimination performed on an $n \times n$ matrix $X = (x_{ij})$ of indeterminates, with pivots chosen on the diagonal. This yields a (sequential) computation of $\det X$ in time $O(n^3)$, using $+$, $-$, $*$, $/$. Strassen (1973) gives a general recipe for avoiding divisions which in our case works as follows: When we execute the above Gaussian elimination on the $n \times n$ identity matrix I, the only divisions that occur are by 1. In other words, if we shift the input matrix X by the negative of the identity matrix and thus consider new indeterminates $y_{ij} = x_{ij} - \delta_{ij}$, then all divisions in the algorithm (with x_{ij} replaced by $y_{ij} + \delta_{ij}$) are by rational functions in the y_{ij}'s which are 1 for $y_{11} = y_{12} = \cdots = y_{nn} = 0$, since this corresponds to setting $X = I$. These rational functions are invertible in the ring $R = F[\![y_{11}, y_{12}, \ldots, y_{nn}]\!]$ of formal power series in the y_{ij}'s. Also, $\det X = \det((y_{ij} + \delta_{ij})_{i,j})$ is a polynomial in R of degree n.

We now modify our Gaussian elimination as follows: Replace each division $f/(1-g)$, where g has constant term 0, by a multiplication $f * (1 + g + g^2 + \cdots)$. Also, instead of computing a power series $h = h_0 + h_1 + \cdots$, where $h_i \in R$ is homogeneous of degree i, compute only the homogeneous parts $h_0, \ldots, h_n$. This can be done at a cost of $\leqslant (n+1)^2$

operations per original operation. All this leads to a sequential computation of $\det X$ in R with sequential time $O(n^5)$, using $1, +, -, *$ only. (This timing estimate can be improved; see Strassen, 1973.) Substituting $y_{ij} = x_{ij} - \delta_{ij}$ in this algorithm, we get a division-free straight-line algorithm in $F[x_{11}, x_{12}, \ldots, x_{nn}]$ that computes $\det X$ in time $O(n^5)$.

We can now apply the parallelization method of Valiant–Skyum–Berkowitz–Rackoff (1981) to obtain a parallel algorithm for the determinant using $O(\log^2 n)$ time and a polynomial number of processors. Neither division nor branching occurs.

Obviously INVERSION and NONSINGULAR EQUATIONS are not harder than DETERMINANT, but they require a division step at the end of the computation. For the characteristic polynomial, we execute the sequential division-free determinant algorithm on $X - tI$. Each step computes a polynomial in t, and we split the step into $\leqslant (n+1)^2$ operations in $F[x_{11}, x_{12}, \ldots, x_{nn}]$ by computing the coefficients of $t^0, t^1, \ldots, t^n$ separately. Parallelization applies again to yield the result. ∎

Remark 1. The above two-step conversion process applies to any sequential computation that computes a polynomial $f \in F[x_1, \ldots, x_m]$ of degree n in time t.

The first step is to get rid of divisions à la Strassen (1973). For this, we have to find values $a_1, \ldots, a_m \in F$ such that $h(a_1, \ldots, a_m) \neq 0$ for each division g/h in the computation. Then we shift the inputs by the negative of the a_i's and thus consider new indeterminates $y_i = x_i - a_i$. Now each division in the algorithm (where every occurrence of x_i is replaced by $y_i + a_i$) is by a rational function in $y_1, \ldots, y_m$ which has a nonzero value for $y_1 = \cdots = y_m = 0$. Such a rational function is invertible in the ring $R = F[\![y_1, \ldots, y_m]\!]$, and $\bar{f} = f(y_1 + a_1, \ldots, y_m + a_m) \in R$ is a polynomial of degree n.

As above, we replace every division by a multiplication in R, and for all operations compute the homogeneous parts of degree $\leqslant n$. This yields a sequential computation in R for $\bar{f}$ with time $O(tn^2)$, using only $+, -, *$, and constants. Back-substituting we get an $O(tn^2)$-algorithm in $F[x_1, \ldots, x_m]$ that computes f without divisions.

The second step now is to apply the parallelization technique of Valiant–Skyum–Berkowitz–Rackoff (1981) to obtain a parallel algorithm with parallel time $O(\log^2(tn))$ using a polynomial (in t and n) number of processors.

How can we find the $a_1, \ldots, a_m$ required in the first step? Every rational function g computed by the given algorithm can be written as a quotient $g = b/c$ of polynomials in $F[x_1, \ldots, x_m]$ with $\deg b, \deg c \leqslant 2^t$. The product d of all such denominators c has degree $\leqslant t2^t$. For every subset $P \subseteq F$ with $\#P > mt2^t$ there exists $a = (a_1, \ldots, a_m) \in P^m$ such that $d(a_1, \ldots, a_m) \neq 0$. Such an a satisfies the requirements.

In fact, there is a Monte Carlo algorithm to find such an a: Take P with $\#P \geqslant 2mt2^t$ and choose $a \in P^m$ at random. Then run the algorithm with input a, and report "failure" if a division by 0 is attempted. By Lemma 1, the probability that this will occur is $\leqslant \frac{1}{2}$. Note that if $F = \mathbb{Q}$, say, and we have a "random bit generator," then we can choose $P = \{-mt2^t, \ldots, mt2^t\}$ and generate the required a's in time polynomial in t and m.

If F is finite, say with p elements, then we can take an irreducible polynomial $u \in F|z|$ of degree l such that $p^l \geqslant 2mt2^t$, and consider the extension field $G = F|z|/(u)$ of F. We can find such a u in random polynomial (in m and t) time (Rabin, 1980). We now view the algorithm as a computation over $G|x_1, \ldots, x_m|$. The field G has $\geqslant 2mt2^t$ elements, so that we can apply the above Monte Carlo procedure to find an appropriate $a \in G^m$. Each operation in G can be simulated by $O(l^2)$ operations in F. Thus the "sequential to parallel" conversion also works over finite fields.

One important property that is lost in the conversion is uniformity: given a uniform family of sequential algorithms, it is not clear how to make the resulting family of parallel algorithms uniform. In order to conserve uniformity, it would be sufficient to know the existence of a "test" polynomial d as above, with degree polynomial (not exponential as the d above) in tn. (The family of parallel algorithms in Theorem 1 is, of course, uniform, because we could explicitly write down $a_1, \ldots, a_m$.)

Remark 2. The algorithm of Theorem 1 uses $O(n^{15})$ processors. Recently, Berkowitz (1982) has given an algorithm that computes $n \times n$ determinants in parallel time $O(\log^2 n)$ using $O(n^{3.5})$ processors.

THEOREM 2. *For any field F,* GCD *can be computed in parallel time* $O(\log^2 n)$.

Proof. Let $f, g \in F|x|$ be nonzero, $\deg f = m \leqslant n = \deg g$. Let $h = \gcd(f, g)$ be the monic greatest common divisor of f and g, and $d = \deg h$. There exist polynomials $u, v \in F|x|$ such that $h = uf + vg$, $\deg u < n - d$, and $\deg v < m - d$. For any $k \geqslant 0$ and polynomials $s = \sum s_i x^i$ and $t = \sum t_i x^i$, the conditions "$sf + tg$ is monic of degree k, and $\deg s < n - k$" translate into the following $(n + m - 2k) \times (n + m - 2k)$ systems S_k of linear equations in the coefficients of s and t:

$$\begin{bmatrix} f_m & & & g_n & & \\ f_{m-1} & f_m & & g_{n-1} & g_n & \\ \vdots & \ddots & & \vdots & \ddots & g_n \\ & & f_m & & & \vdots \\ f_0 & & \vdots & & & \\ & \ddots & & g_0 & & \\ & & f_0 \cdots f_k & & g_0 \cdots & g_k \end{bmatrix} \cdot \begin{bmatrix} s_{n-k-1} \\ \vdots \\ s_0 \\ t_{m-k-1} \\ \vdots \\ t_0 \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}.$$

So for f, g as above, the following algorithm computes $\gcd(f, g)$ in parallel time $O(\log^2 n)$:

(1) Compute in parallel $a_0, \ldots, a_m$, where $a_k = \det A_k$ and A_k is the coefficient matrix of S_k.

(2) Set $d = \min\{k: a_k \neq 0\}$.

(3) Compute a solution (s, t) of S_d. (Note that S_d is a nonsingular system.)

(4) Compute $\gcd(f, g) = sf + tg$. ∎

It would be important to have a similar result for the gcd of two integers, and we ask the

Open Question 1. Is INTEGER GCD $\in NC$?

4. Rank of Matrices

For algorithms computing the rank of matrices, i.e., the maximal size of nonsingular submatrices, we can restrict attention to square matrices (by padding with zeroes if necessary). We denote by $M_n(S)$ the set of $n \times n$ matrices with entries from a set S. The rank cannot in general be considered as an element of the ground field, and we have to make some output convention such as the following: we require the algorithm to compute $f_1, \ldots, f_n \in F(x_{11}, x_{12}, \ldots, x_{nn})$ such that for any $A \in M_n(F)$

$$\operatorname{rank}(A) = \max\{i: i = 0 \text{ or } f_i(A) \neq 0\}.$$

We remark that this convention is inadequate over $\mathbb{C}$, since then the n equations $f_1(A) = \cdots = f_n(A) = 0$ in n^2 variables should have only $A = 0$ as solution. If $n > 1$, then this is impossible over any algebraically closed field.

Ibarra–Moran–Rosier (1980) have the following result: If F is a subfield of $\mathbb{R}$, then $\operatorname{rank}(A) = \operatorname{rank}(A^t A)$ and the matrix $A^t A$ is symmetric and diagonalizable. Hence its rank can be read off from the coefficients of its characteristic polynomial $c = \sum_{0 \leqslant i \leqslant n} c_i x^i$ as

$$\operatorname{rank}(A) = \max\{j: c_{n-j} \neq 0\}.$$

Thus RANK can be computed in parallel time $O(\log^2 n)$ for such a field. They also show that this is true if F is a subfield of $\mathbb{C}$ and one is allowed to use complex conjugation in the algorithm.

The rank question has some nice applications in combinatorial complexity. Consider a bipartite graph G on $2n$ nodes. We associate to it an $n \times n$ matrix X over $\mathbb{Q}(x_{11}, x_{12}, \ldots, x_{nn})$ which has x_{ij} in the (i, j) position if node i is connected to node j, and zero otherwise. Then the maximal size of a

matching in G is equal to $\operatorname{rank}(X)$ (Edmonds, 1967). By substituting randomly chosen integers (from a fixed finite range, see Lemma 1) for the indeterminates and computing the rank of such matrices over $\mathbb{Q}$, we get a Monte Carlo algorithm to determine the maximal size of matchings in G, and hence a (nonuniform) algorithm for this problem whose parallel time is $O(\log^2 n)$. It would be interesting to have an algorithm of this type that actually constructs a maximal matching.

Next consider a directed graph with each edge being assigned an integer capacity that is polynomially bounded by the number of nodes of the graph. Feather (1981) reduces the problem of finding the maximum flow in such a graph to the above maximal matching problem, and thus obtains an $O(\log^2 n)$ algorithm.

It is interesting to note that without the restriction on the capacities the problem is log space complete for P (Goldschlager–Shaw–Staples, 1982), and hence we do not expect it to have a solution in parallel time $O(\log^k n)$ for some k.

The rank algorithm by Ibarra–Moran–Rosier provides a nice solution for fields like $\mathbb{Q}$ and $\mathbb{R}$. For the general case, in particular for finite fields, we have the following result:

THEOREM 3. *For any field F, we can compute the rank of $n \times n$ matrices over F with a Monte Carlo algorithm with error probability $\leqslant 0.95$ using parallel time $O(\log^2 n)$ and a polynomially bounded total number of processors. No division occurs.*

Proof. Let F be a field, $n \in \mathbb{N}$ and $A \in M_n(F)$. We assume some finite subset $P \subseteq F$ of cardinality p such that either $p \geqslant 3n^2$ or $P = F$. Furthermore, we assume a "random element generator" for P that produces in one step of a processor a randomly chosen element from P with respect to the uniform distribution on P. (Thus if $\#F \geqslant 3n^2$, we can take any large enough subset for P, and otherwise we take $P = F$.)

For a matrix $M \in M_n(F)$ and $1 \leqslant i \leqslant n$ let $P_i(M)$ be the principal $i \times i$ minor of M, i.e., the square submatrix of M consisting of the first i rows and columns of M, and $p_i(M) = \det(P_i(M))$. We perform the following algorithm to compute $\operatorname{rank}(A)$:

(1) Choose two matrices B, $C \in M_n(P)$ at random.

(2) For all i, $1 \leqslant i \leqslant n$, compute $f_i = p_i(BAC)$.

The output s can be read off in the manner described above:

$$s = \max\{i: i = 0 \text{ or } f_i \neq 0\}.$$

It is clear from Theorem 1 that this algorithm uses parallel time $O(\log^2 n)$, a polynomially bounded number of processors, and no division. Also

$s \leqslant \text{rank}(A)$. The remainder of this proof is devoted to establishing the estimate

$$\text{Prob}(s = \text{rank}(A)) \geqslant 0.05.$$

Let $r = \text{rank}(A)$. We consider

$$g_A = p_r(QAR) \in F[Q_{11}, Q_{12}, \ldots, R_{nn}]$$

as a polynomial in the indeterminate entries Q_{ij}, R_{ij} of the $n \times n$-matrices Q, R. Thus g_A is linear in each of its $2n^2$ indeterminates. It is nonzero, since by permuting the rows and columns of A one can obtain a matrix with nonsingular principal $r \times r$ minor; in other words, there exist permutation matrices S, $T \in M_n(F)$ such that $P_r(SAT)$ is nonsingular, and hence $g_A(S, T) \neq 0$. We also have

$$\begin{aligned} \text{Prob}(s < r) &= \text{Prob}(g_A(B, C) = 0) \\ &= p^{-2n^2} \#\{(B, C) \in M_n(P)^2 : g_A(B, C) = 0\}. \end{aligned}$$

It remains to show that

$$\text{Prob}(g_A(B, C) = 0) \leqslant 0.95.$$

We now distinguish two cases.

Case 1. $p \geqslant 3n^2$. By Lemma 1,

$$\text{Prob}(g_A(B, C) = 0) \leqslant 2n^2/p \leqslant 0.95.$$

Case 2. $P = F$. Let

$$D = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix} \in M_n(F)$$

be a diagonal matrix with $\text{rank}(D) = r$. Then

$$P_r(BDC) = P_r(B)\, P_r(C), \qquad p_r(BDC) = p_r(B)\, p_r(C).$$

By Lemma 2(ii), we have

$$\text{Prob}(p_r(B) = 0) < \tfrac{3}{4},$$

and similarly for C. Since the entries of B and C are chosen independently, $p_r(B)$ and $p_r(C)$ are independent random variables. Hence we get

$$\begin{aligned}\text{Prob}(p_r(BDC)=0) &= \text{Prob}(p_r(B)\,p_r(C)=0)\\ &= \text{Prob}(p_r(B)=0)+\text{Prob}(p_r(C)=0)\\ &\quad -\text{Prob}(p_r(B)=p_r(C)=0)\leqslant \tfrac{15}{16}<0.95,\end{aligned}$$

using the monotonicity of the function $2x-x^2$ for $0\leqslant x\leqslant 1$.

Getting back to our input matrix A, we note that there exist nonsingular matrices S, $T\in M_n(F)$ such that $A=SDT$. Then

$$\begin{aligned}&\text{Prob}(g_A(B,C)=0)=\text{Prob}(p_r(BAC)=0)\\ &\quad=\text{Prob}(p_r(BSDTC)=0)=\text{Prob}(p_r(BDC)=0)\leqslant 0.95.\end{aligned}$$

The third equality follows from the fact that the mapping

$$\begin{aligned}M_n(F)\times M_n(F)&\to M_n(F)\times M_n(F)\\ (B,C)&\to (BS^{-1},T^{-1}C)\end{aligned}$$

is a bijection. ∎

For the two lemmas that establish the probabilistic estimates, we assume the following set-up: A field F, $P\subseteq F$ finite with $p\geqslant 2$ elements, and $d\in\mathbb{N}$.

LEMMA 1. *Let $m\in\mathbb{N}$, $f\in F[x_1,\ldots,x_m]$ be nonzero and of degree $\leqslant d$ in each variable. Then*

$$\text{Prob}(f=0)\leqslant md/p.$$

Proof. Let

$$q=\text{Prob}(f=0)=p^{-m}\#\{a\in P^m: f(a)=0\}.$$

We show by induction on m that $q\leqslant md/p$. The case $m=1$ is obvious. For $m>1$ we can assume that x_m occurs in f (otherwise, apply the induction hypothesis), and write f as a polynomial in x_m

$$f=\sum_{0\leqslant i\leqslant e} g_i x_m^i,\qquad g_0,\ldots,g_e\in F[x_1,\ldots,x_{m-1}],\quad g_e\neq 0,\quad 0<e\leqslant d.$$

Then

$$\begin{aligned}\#\{a\in P^m: f(a)=0\}&\leqslant \#\{a\in P^m: g_e(a)=0\}\\ &\quad+\#\left\{a\in P^m: g_e(a)\neq 0 \text{ and } \sum_i g_i(a)\,a_m^i=0\right\}\\ &\leqslant p^m(m-1)d/p+dp^{m-1}=p^m md/p.\end{aligned}$$

Hence $q\leqslant md/p$. ∎

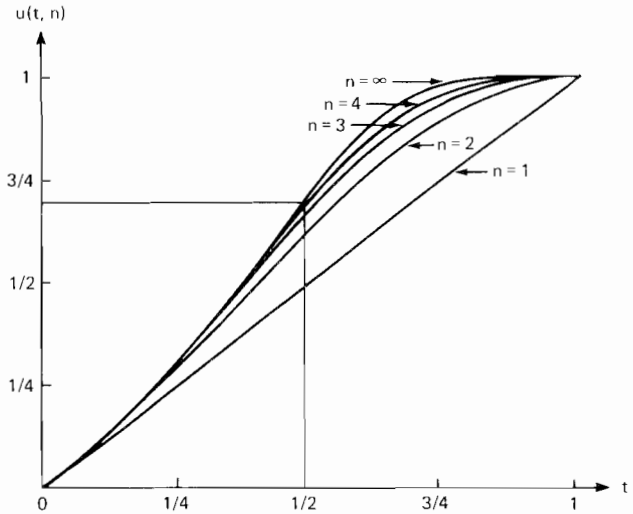

FIG. 1. The upper bound $u(t, n)$ on the probability that an $n \times n$ matrix is singular. The entries of the matrix are nonconstant polynomials of degree $\leqslant d$ over an arbitrary field F whose arguments are randomly chosen from a subset of F with p elements, and $t = d/p$. The values $n = 1, 2, 3, 4$ are shown, and $n = \infty$ shows an upper bound for any $n \in \mathbb{N}$.

Lemma 2 gives a sharp estimate of the probability that a matrix of polynomials is singular. We consider indeterminates $x_{11}, x_{12}, \ldots, x_{nn}$ over F, and an $n \times n$ matrix g of univariate polynomials $g_{ij} \in F[x_{ij}]$ of degree $\leqslant d$. We assume that no g_{ij} is constant. Furthermore, we introduce the function $u(t, n) = 1 - \prod_{1 \leqslant i \leqslant n} (1 - t^i)$ for $0 \leqslant t \leqslant 1$ and $n \in \mathbb{N}$. u is monotonically increasing in both arguments (see Fig. 1).

LEMMA 2. *In the above set-up, let q be the probability that* $\det(g)$ *is zero. Then*

(i) $q \leqslant u(d/p, n)$,

(ii) *if* $d = 1$, *then* $q < \frac{3}{4}$,

(iii) *if* $d = 1$ *and P is a field, then* $q = u(1/p, n)$.

Proof. Let $t = d/p$.

(i) We have to show that

$$q = p^{-n^2} \# \{A \in M_n(P) : \det(g(A)) = 0\} \leqslant u(t, n),$$

where $g(A)$ is the matrix with (i, j) entry $g_{ij}(A_{ij})$. For $1 \leqslant r \leqslant n$, let T_r be the set of all $r \times n$ matrices with entries from P, and $S_r = \{A \in T_r : \operatorname{rank}(g(A)) = r\}$. Now let $A \in S_{r-1}$ and $I \subseteq \{1, \ldots, n\}$ such that $\# I = r - 1$

and the square minor of $g(A)$ with rows $1,\ldots, r-1$ and columns $i \in I$ is nonsingular. A vector $y \in F^n$ which is linearly dependent on the row vectors of $g(A)$ is determined by its entries y_i for $i \in I$.

For a vector $z \in T_1$ such that $y = (g_{r1}(z_1),\ldots, g_{rn}(z_n))$ is linearly dependent on the rows of $g(A)$, we can in general choose the entries z_i for $i \in I$ arbitrarily; for the other entries we then have at most d choices each, determined by the fixed value for $g_{ri}(z_i)$. In particular, there are at most $p^{r-1}d^{n-r+1}$ such vectors $z \in T_1$. Thus

$$\#S_r \geqslant (p^n - p^{r-1}d^{n-r+1})\,\#S_{r-1} = p^n(1 - t^{n-r+1})\,\#S_{r-1},$$

and by induction on r we get

$$\#S_r \geqslant p^{rn} \prod_{n-r<i\leqslant n} (1 - t^i).$$

Thus

$$q = p^{-n^2}(p^{n^2} - \#S_n) \leqslant 1 - \prod_{1\leqslant i\leqslant n} (1 - t^i) = u(t, n).$$

(ii) For $0 \leqslant t \leqslant 1$, let

$$v(t) = \lim_{n\to\infty} u(t, n) = 1 - \prod_{1\leqslant i} (1 - t^i).$$

The product in v is uniformly convergent on every interval $[0, a]$ with $a < 1$ (see Henrici (1977), Sect. 8.2, for a discussion and the relation of v with combinatorial questions), and v is a monotonically increasing function. In particular, for $0 \leqslant t \leqslant \frac{1}{2}$ and $n \in \mathbb{N}$ we have

$$0 \leqslant u(t, n) < v(t) \leqslant v(\tfrac{1}{2}) = 0.7112\ldots < \tfrac{3}{4}.$$

This proves (ii), since then $t = 1/p \leqslant \frac{1}{2}$.

(iii) Under the hypotheses of (iii), we can replace "at most $p^{r-1}d^{n-r+1}$" by "exactly p^{r-1}" in the argument for (i), and

$$\#S_r = p^{rn} \prod_{n-r<i\leqslant n} (1 - t^i)$$

follows. This implies $q = u(t, n)$. This fact has been discovered several times, see, e.g., Jordan (1911), Fine–Niven (1944), and Rotman (1973, Theorem 8.8). ∎

Statement (iii) shows that the estimate in (i) is sharp in a certain sense. We remark that the estimate of (i) also holds when the determinant is replaced by the permanent; the proof is slightly different.

It is now clear what the estimates in the proof of Theorem 3 really should

be: if $c = 2v(\frac{1}{2}) - v(\frac{1}{2})^2 = 0.9166...$, then we should assume either $p \geqslant 2n^2/c$ or $P = F$, and the error probability then is $\leqslant c$.

When interpreted as the decision problem "is A in $U_r = \{B \in M_n(F)$: $\operatorname{rank}(B) \leqslant r\}$?", the above Monte Carlo algorithm always answers correctly if $A \in U_r$, and correctly with probability $\geqslant \frac{1}{20}$ if $A \notin U_r$. We can improve this behaviour to get a Las Vegas algorithm that always answers correctly and with high probability has a low parallel running time.

THEOREM 4. *For any field F, there exists a Las Vegas algorithm for the rank of $n \times n$ matrices that uses parallel time $O(\log^2 n)$, no divisions and a polynomial number of processors and that either returns the rank of the input matrix (and a maximal nonsingular minor) or reports "failure." The probability of the second case is $\leqslant 2^{-n}$.*

Proof. Let $A \in M_n(F)$. For $1 \leqslant i \leqslant n$ let $A_i \in M_n(F)$ consist of the first i rows of A and zero rows otherwise, and $r_i = \operatorname{rank}(A_i)$. Apply the algorithm of Theorem 3 independently $28n$ times in parallel to each A_i, and let s_i be the maximum of the numbers computed for A_i. Thus $s_i \leqslant r_i$, and for each i

$$\operatorname{Prob}(s_i < r_i) \leqslant (\tfrac{19}{20})^{28n}.$$

Let $I = \{i: 1 \leqslant i \leqslant n$ and $s_{i-1} < s_i\}$ (with $s_0 = 0$), and $A' \in M_n(F)$ be the matrix whose ith row is the ith row of A if $i \in I$ and zero otherwise. Perform the computation that was done above for the rows of A now for the columns of A' to obtain a set $J \subseteq \{1,..., n\}$. The parallel time for all this is $O(\log^2 n)$, and

$$\operatorname{Prob}(\text{the } I \times J \text{ minor of } A \text{ is not a maximal (square) nonsingular minor}) \leqslant 2n(\tfrac{19}{20})^{28n} \leqslant 2^{-n}.$$

If $\#I \neq \#J$, then report failure. Else apply the determinant algorithm of Theorem 1 to compute A_{IJ} and each A_{KL} with $I \subseteq K$, $J \subseteq L$, $\#K = \#L = \#I + 1$ in parallel time $O(\log^2 n)$. Here A_{KL} denotes the determinant of the minor of A with rows from K and columns from L. If either $A_{IJ} = 0$ or some such A_{KL} is nonzero, then report failure. Otherwise, return $s_n = \operatorname{rank}(A)$ and the $I \times J$ minor as a maximal nonsingular minor. ∎

Using the rank algorithm over $\mathbb{R}$ by Ibarra–Moran–Rosier and the above construction, we get a deterministic algorithm that uses parallel time $O(\log^2 n)$ and computes a maximal nonsingular minor. Can we have similar improvement for arbitrary fields? Of course our Monte Carlo algorithm yields nonuniform algorithms using parallel time $O(\log^2 n)$. It would be particularly interesting to have an answer to the following:

Open Question 2. Can RANK be computed in parallel time $O(\log^2 n)$ over finite fields, using a uniform family of deterministic algorithms?

In connection with the maximal matching problem we remark that our algorithm gives an $O(\log^2 n)$ method to determine a set of nodes on which a matching of maximal size will take place; not, however, the actual matching.

5. Reductions

We have considered a number of algebraic problems whose parallel complexity is now known to be $O(\log^2 n)$. Whether or not the parallel complexity for any of these (and some closely related) problems can be reduced to $O(\log n)$ remains a fundamental challenge for all of complexity theory (see, for example, Borodin, 1982, and Valiant, 1982). It is natural then to study the relative complexity of these problems.

Valiant (1982) introduced the notion of algebraic projections as a strong type of reducibility between polynomials. For functions $P = (P(n))$ and $P' = (P'(n))$, we write $P \lesssim P'$ to informally denote the fact that P can be reduced to P' essentially by multiple use of projections and possibly some simple (i.e., of $O(\log n)$ parallel complexity) arithmetic operations. A precise definition for $\lesssim$ will not be developed here. Given any reasonable definition of reducibility $P \lesssim P'$, and certainly for the specific reductions given in this section, it follows that $T(P)$ and $T(P')$ satisfy $T(P) = O(T(P'))$, using the fact that for all interesting functions P, $\log n = O(T(P(n)))$. We write $P \approx P'$ to denote $P \lesssim P'$ and $P' \lesssim P$, and we write $P \lesssim P' + P''$ to denote that both P' and P'' are used in the reduction.

Csanky (1976) has the following reductions:

(a) Over any field,

$$\text{INVERSION} \approx \text{NONSINGULAR EQUATIONS}$$
$$\leqslant \text{DETERMINANT} \leqslant \text{CHARACTERISTIC POLYNOMIAL}.$$

(b) The characteristic polynomial can be computed by evaluating it at several points (using a determinant algorithm) and interpolating. If the field contains the necessary roots of unity to support a fast Fourier transform, then the interpolation can be performed in $O(\log n)$, using the roots of unity as interpolation points. We then have

$$\text{CHARACTERISTIC POLYNOMIAL} \lesssim \text{DETERMINANT}.$$

(c) If the field has characteristic zero, then

$$\text{DETERMINANT} \lesssim \text{NONSINGULAR EQUATIONS}.$$

Thus, over $\mathbb{C}$ all four problems are equivalent.

Next, we consider the problems BASIS (=computing a maximal linearly independent subset of a given set of vectors), EQUATIONS (=deciding

whether a (possibly singular) system of linear equations has a solution, and in the affirmative case, computing one solution), and NULLSPACE (=computing a basis for the nullspace of a matrix).

THEOREM 5. *For an arbitrary field, we have the following reductions*:

(i) BASIS $\approx$ RANK $\lesssim$ NULLSPACE $\lesssim$ RANK + INVERSION,

(ii) EQUATIONS $\lesssim$ RANK + INVERSION.

Proof. (i) We note that the rank is implied directly by BASIS or NULLSPACE. The reduction BASIS $\lesssim$ RANK is obtained by including v_i in the basis iff $\operatorname{rank}(v_1,\ldots,v_{i-1}) < \operatorname{rank}(v_1,\ldots,v_i)$. For the remaining reduction, we can compute a maximal nonsingular minor M for an input matrix A, by applying BASIS (say) first to the columns of A and then to the rows of the selected columns. Without loss of generality let M be the upper $r \times r$ submatrix of A. For each i satisfying $r < i \leqslant n$, we can solve the nonsingular system $Mx_i = y_i$, where y_i denotes the first r rows of the ith column of A. A basis for the nullspace of A then consists of the vectors

$$\begin{bmatrix} x_i \\ 0 \\ \vdots \\ -1 \\ \vdots \\ 0 \end{bmatrix}, \qquad r < i \leqslant n,$$

that is, x_i is extended by 0's except for an -1 in the ith position.

(ii) For EQUATIONS, we again compute a maximal nonsingular minor M of the input matrix, solve the corresponding nonsingular system of linear equations, set all indeterminates not corresponding to columns of M equal to zero, and check whether this constitutes a solution. If it does not, then the input system has no solution at all. ∎

Allowing nonuniform reductions, and allowing the concept of reducibility to use linear transformations, it follows (by Theorem 3 and Csanky's reductions) that RANK $\lesssim$ CHARACTERISTIC POLYNOMIAL $\lesssim$ DETERMINANT and hence EQUATIONS $\lesssim$ DETERMINANT. We also know (from Theorem 2) that GCD $\lesssim$ DETERMINANT.

We are left with a number of potential reductions which are either completely unresolved or hold only in the nonuniform case. In particular, we ask the following:

Open Question 3. (a) Are any of the above problems $\lesssim$GCD?

(b) Is DETERMINANT $\lesssim$ NONSINGULAR EQUATIONS for arbitrary fields?

The latter question is also open in the sequential setting (see Baur–Strassen, 1983).

6. Conclusions

We have laid the foundation for what might be called a "theoretical package for parallel symbolic manipulation." The problems investigated include gcd of polynomials, solution of linear equations, determinant and rank of matrices. They all can be solved in parallel time $O(\log^2(\text{input size}))$; for rank a Las Vegas method is used.

Further important routines for this "theoretical package" include factoring polynomials over finite fields (which provided the original motivation for this paper; consider the critical role of the GCD and NULLSPACE in the Berlekamp (1970) factoring algorithm), continued fractions and partial fraction expansion of rational functions, Padé approximation of power series, and interpolation. They will be discussed in a forthcoming paper (von zur Gathen, 1983).

Finally, as in sequential computation, we remark that integer problems are often more difficult to understand than the corresponding polynomial problem. For example, it remains an open problem as to whether or not a fast (e.g., $O(\log^2 n)$) parallel algorithm exists for the gcd of two n-bit numbers. An even more challenging open problem concerns the parallel complexity of determining primality.

Acknowledgments

We are very grateful to Steve Cook for an observation which led to Theorem 2, to both Steve Cook and Les Valiant for pointing out relations with combinatorial problems, and to Volker Strassen for an improvement in the algorithm for Theorem 3. Thanks also go to Yaacov Yesha for pointing out one of the references.

References

Bareiss, E. H. (1968), Sylvester's identity and multistep integer-preserving Gaussian elimination, *Math. Comput.* **22**, 565–578

Baur, W. and Strassen, V. (1983), The computational complexity of partial derivatives, *Theor. Comput. Sci.* **22**, 317–330.

Berlekamp, E. R. (1970), Factoring polynomials over large finite fields, *Math. Comput.* **24**, 713–735.

Berkowitz, S. J. (1982), On computing the determinant in small parallel time using a small number of processors, manuscript, October 1982.

BORODIN, A. (1982), Structured vs. general models in computational complexity, *in* "Logic and Algorithmic, Symposium in Honour of Ernst Specker," Monographie No. 30 de l'Enseignement Mathématique, pp. 47–65.

CSANKY, L. (1976), Fast parallel matrix inversion algorithms, *SIAM J. Comput.* **5**, 618–623.

EDMONDS, J. (1967), Systems of distinct representatives and linear algebra, *J. Res. Nat. Bur. Stand.* **71B** (4), 241–245.

FEATHER, T. (1981), Private communication, November 1981.

FINE, N. J. AND NIVEN, I. (1944), The probability that a determinant be congruent to a (mod m), *Bull. Amer. Math. Soc.* **50**, 89–93.

FORTUNE, S. AND WYLLIE, J. (1978), Parallelism in random access machines, *in* "Proc. 10th ACM Symposium on Theory of Computing," pp. 114–118, San Diego, California.

GATHEN, J. VON ZUR (1983), Parallel algorithms for algebraic problems, *in* "Proc. 15th ACM Symp. Theory of Computing, Boston."

GOLDSCHLAGER, L. M., SHAW, R. A., AND STAPLES, J. (1982), The maximum flow problem is log space complete for P, *Theor. Comput. Sci.* **21**, 105–111.

HENRICI, P. (1977), "Applied and Computational Complex Analysis," Vol. 2, Wiley, New York.

IBARRA, O., MORAN, S., AND ROSIER, L. E. (1980), A note on the parallel complexity of computing the rank of order n matrices, *Inf. Process. Lett.* **11**, 162.

JORDAN, C. (1911), Sur le nombre des solutions de la congruence $|a_{ik}| \equiv A$ (mod M), *J. Math. Pures Appl.* (6), 7, 409–416.

PIPPENGER, N. (1979), On simultaneous resource bounds, *in* "Proc. IEEE 20th Annual FOCS," October 1979, 307–311.

RABIN, M. O. (1980), Probabilistic algorithms in finite fields, *SIAM J. Comput.* **9**, 273–280.

ROTMAN, J. J. (1973), "The Theory of Groups," (2nd ed.), Allyn & Bacon, Boston.

STRASSEN, V. (1973), Vermeidung von Divisionen, *J. Reine Angew. Math.* **264**, 184–202.

VALIANT, L., SKYUM, S., BERKOWITZ, S., AND RACKOFF, C. (1981), Fast parallel computation of polynomials using few processors, *SIAM J. Comput.*, to appear.

VALIANT, L. (1982), Reducibility by algebraic projections, *in* "Logic and Algorithmic, Symposium in Honour of Ernst Specker," Monographie No. 30 de l'Enseignement Mathématique, pp. 365–380.

Printed by the St. Catherine Press Ltd., Tempelhof 41, Bruges, Belgium