AD-A113 577 CORNELL UNIV ITHACA NY DEPT OF COMPUTER SCIENCE F/G 12/1
FAST PARALLEL MATRIX AND GCD COMPUTATIONS.(U)
APR 82 A BORODIN, J VON ZUR GATHEN N00014-76-C-018
UNCLASSIFIED CU-CSD-TR-82-487 NL

1 OF 1
AD
A113577

END
DATE
FILMED
5-82
DTIC

AD A113577

82 04 19 007

12

FAST PARALLEL MATRIX AND GCD COMPUTATIONS

Allan Borodin*
Joachim von zur Gathen*
John Hopcroft**

TR 82-487

April 1982

* Department of Computer Science, University of Toronto, Toronto, Canada

** Department of Computer Science, Cornell University, Ithaca, New York

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.

Abstract

We present parallel algorithms to compute the determinant and characteristic polynomial of $n \times n$-matrices and the gcd of polynomials of degree $\leq n$. The algorithms use parallel time $O(\log^2 n)$ and a polynomial number of processors. We also give a fast parallel Monte Carlo algorithm for the rank of matrices. All algorithms work over arbitrary fields.

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.). Clearly 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)/# 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 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 [76] 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 [81].

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. That seems to be a slightly more difficult question. We present a fast parallel Monte Carlo 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 computation such as the PRAM (Fortune-Wyllie [78]), with arithmetic and tests in F as basic operations. The algorithms all use $O(\log^2 n)$ parallel time and $n^{O(1)}$ processors when n is the number of inputs. In particular, it follows that the determinant and gcd problems are in the appropriate analogue of uniform NC (Pippenger [79]), and the rank problem is in the Monte Carlo or non-uniform analogue of NC.

When the ground field F is Q or a finite field, one 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 non-uniform Boolean version of NC (in fact, for F=Q in NC). The essential point for this is that (for F=Q) according to Edmonds [67] (see also Bareiss [68]) the intermediate values in Gaussian elimination are reasonably small.

The most appropriate model for our algorithms seems to be a parallel version of algebraic computation trees (Strassen [81]). Thus at each node a number of operations/tests can be performed; the maximal such number is the number of processors of the parallel algebraic computation tree. If the input space is F^n and the computation is defined at all inputs, we get a partition $S = (S_1, \ldots, S_k)$ of F^n by forming for each output leaf the set consisting of the inputs for which the computation terminates at that leaf. On each S_i, a sequence $f_i = (f_{i1},\ldots,f_{i\ell})$ of rational functions is computed. Such an object (f,S) with $f=(f_1,\ldots,f_k)$ is called a collection; see Strassen [81].

The fast parallel algorithms that we seek should have three properties: small parallel time, small number of processors, small size. More precisely, the parallel time should be polynomial in logn, the number of processors and the number of nodes of the tree should be polynomial in n where n is the number of inputs. The algorithms that we present have these properties, in particular time $O(\log^2 n)$.

The basic steppingstone for the whole theory is the result by Valiant-Skyum-Berkowitz-Rackoff [81]. It says that any sequential program computing a polynomial of degree $\leq n$ 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 $\leq n$).

The collection for INVERSION(n) consists of the sets $S_1, S_2 \subseteq F^{n^2}$, where $S_1 = \{(a_{ij}) \in F^{n^2} : \det(a_{ij})=0\}$ and $S_2 = F^{n^2} \setminus S_1$, and the output functions are $f=0$ on S_1 (signalling that the input matrix is not invertible) and $f = (f_{11}, \ldots, f_{nn})$ with $f_{ij} \in F(x_{11}, \ldots, x_{nn})$ the (i,j)-entry of the inverse of the $n \times n$-matrix (x_{ij}). The collections for the first three problems are similarly obvious, and one for GCD can be found in Strassen [81]. We write, e.g. INVERSION for the sequence (INVERSION(n))

The following result was proved by Csanky [76], but only for fields of characteristic zero.

Theorem 1 For any field F, DETERMINANT, INVERSION, NONSINGULAR EQUATIONS, CHARACTERISTIC POLYNOMIAL can be computed in parallel time $O(\log^2 n)$ using a polynomially bounded number of processors. Branching does not occur, and for DETERMINANT and CHARACTERISTIC POLYNOMIAL no divisions are necessary.

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 $O(n^3)$ (Coppersmith-Winograd [81]: $O(n^{2.496})$). When we execute this algorithm on the $n \times n$-identity matrix, the only divisions that occur are by 1. According to Strassen [73] we can shift the indeterminates x_{ij} by δ_{ij} and obtain an algorithm for det X without division using time $O(n^5)$. We can now apply the parallelization method of Valiant-Skyum-Berkowitz-Rackoff [81] 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 $\leq (n+1)^2$ 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. $\square$

Remark. The above process of getting rid of divisions and converting to a parallel computation of cost $O(\log n(\log n + \log t))$ applies in principle to any sequential computation that computes a polynomial of degree $\leq n$ in time $\leq t$. In avoiding divisions, one has to shift the indeterminates so that only divisions by non-zero field elements occur if the indeterminates are set to zero. Since it might not always be clear how to pick the constants required for this shift, the conversion may become non-uniform. This is particularly evident over finite fields, where one might have to make a (finite algebraic) field extension.

<u>Theorem 2</u> For any field F, GCD can be computed in parallel time $O(\log^2 n)$.

<u>Proof</u> Let $f,g \in F[x]$ be non-zero, $\deg f = m \le n = \deg g$. If f is not a constant multiple of g, then

$$\deg \gcd(f,g) = \min\{i \in \mathbb{N} : \exists s,t \in F[x],\ \deg s < n-i \text{ and } \deg(sf+tg) = i\}.$$

The latter condition translates into the following $(n+m-2i)\times(n+m-2i)$-system S_i of linear equations:

$$\begin{bmatrix} f_m & & & g_n & & \\ f_{m-1} & f_m & & \vdots & \ddots & \\ \vdots & \ddots & f_m & \vdots & & \ddots \\ \vdots & & & & & g_n \\ f_0 & & \vdots & g_0 & & \vdots \\ & \ddots & \vdots & & \ddots & \vdots \\ & & f_0 \cdots f_i & & & g_0 \cdots g_i \end{bmatrix} \cdot \begin{bmatrix} s_{n-i-1} \\ \vdots \\ s_0 \\ t_{m-i-1} \\ \vdots \\ t_0 \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

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

1. Compute in parallel $a_0,\dots,a_m$, where $a_i = \det A_i$ and A_i is the coefficient matrix of S_i.
2. Set $d = \min\{i : a_i \neq 0\}$.
3. Compute a solution (s,t) of S_d. (Note that S_d is a non-singular system.)
4. Compute $\gcd(f,g) = sf+tg$. $\square$

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). 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 $A \in M_n(F)$

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

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

If $c(A) = \det(A-tI) = \sum_{0 \leq i \leq n} c_i(A) t^i \in F[t]$ is the characteristic polynomial of $A \in M_n(F)$, let us call

$$\operatorname{rank}_{alg}(A) = \max\{i : c_{n-i}(A) \neq 0\}$$

the algebraic rank of A. The relation with rank(A) is described by

$$\begin{aligned} \operatorname{rank}(A) &= n - \dim(\text{nullspace of } A) \\ &= n - \dim(\text{eigenspace of } A \text{ for eigenvalue } 0); \\ \operatorname{rank}_{alg}(A) &= n - \text{order of } c(A) \\ &= n - \text{multiplicity of eigenvalue } 0 \text{ in } c(A). \end{aligned}$$

We thus have $\text{rank}_{\text{alg}}(A) \leq \text{rank}(A)$ for any matrix A.
$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$ is an example where $\text{rank}_{\text{alg}}(A) < \text{rank}(A)$.

This is, however, an "exceptional case": within the set U_r of matrices $A \in M_n(F)$ with $\text{rank}(A) \leq r$, the subset V_r of matrices with $\text{rank}_{\text{alg}}(A) < r$ is described by the nontrivial polynomial condition "$c_{n-r}(A) = 0$". Thus, "most" A in U_r will not be in V_r. E.g. if F is algebraically closed, then V_r has dimension strictly less than the irreducible variety U_r.

Now it follows from Theorem 1 that $\text{rank}_{\text{alg}}(A)$ can be computed fast in parallel, and hence also rank(A) provided $\text{rank}(A) = \text{rank}_{\text{alg}}(A)$. In the sequel we look for ways of obtaining the latter condition.

Ibarra-Moran-Rosier [80] have the following result:
If F is a subfield of $\mathbb{R}$, then $\text{rank}(A) = \text{rank}(A^tA) = \text{rank}_{\text{alg}}(A^tA)$. Hence 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×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 rank(X) (Edmonds[67]). By substituting randomly chosen integers (from a fixed finite range, see Lemma 1 below) 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 (non-uniform) 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 [81] 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 [81]), 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, one can compute the rank of $n \times n$-matrices over F with a Monte Carlo algorithm with error probability $\leq 3/4$ using parallel time $O(\log^2 n)$ and a polynomially bounded total number of processors. Neither division nor branching occur.

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 $p \geq \frac{4}{3}n$ 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 $\#F \geq \frac{4}{3}n$, we can take any large enough subset for P; the proof will be slightly easier in this case. Otherwise, we can either take $P=F$, or compute in parallel time $O(\log n)$ a large enough finite algebraic extension field of F to reduce to the first case.)

We perform the following algorithm to compute rank A:

1. Choose a random $B \in M_n(P)$.
2. Compute $s = \text{rank}_{alg}(AB)$.

It is clear from Theorem 1 that this algorithm uses parallel time $O(\log^2 n)$, a polynomially bounded number of processors, and neither division nor branching. Also $s \leq \text{rank}(A)$. The remainder of this proof is devoted to establishing the estimate

$$\text{Prob}(\{s=\text{rank}(A)\}) \geq 1/4.$$

Let $N = \{1,\ldots,n\}$, $r \in N$, $m = \binom{n}{r}$ and $\binom{N}{r} = \{I \subseteq N: \#I=r\}$. For any $n\times n$-matrix M and $I,J \in \binom{N}{r}$ we write M_{IJ} for the determinant of the $I\times J$-minor of M.

The idea of the proof is the following. $\text{Rank}(A) \geq r$ iff the $m\times m$-array of all minors A_{IJ} has a nonzero entry. Of course it is too expensive to compute this whole array, but a piece of information, namely the sum of the diagonal entries, is easy to compute. Introducing a randomly chosen matrix, it turns out that one has a good chance that this piece of information is sufficient.

A convenient tool for representing the array is the exterior power $\Lambda^r(F^n)$ (for definitions, see e.g. Greub [78]) which is an m-dimensional vector space over F with a basis indexed by $\binom{N}{r}$. We have a natural mapping

$$\Lambda^r: M_n(F) \to \text{End}(\Lambda^r(F^n)) \cong M_m(F)$$

which maps a matrix B to the m×m-array of its r×r-minors, i.e. $(\Lambda^r(B))_{IJ} = B_{IJ}$ for $I,J \in \binom{N}{r}$. Λ^r is multiplicative (Lagrange's identity, see Greub [78]), and obviously $\Lambda^r(\mathrm{id}) = \mathrm{id}$. For any $B \in M_n(F)$ the coefficients of the characteristic polynomial can be expressed as

$$c_{n-r}(B) = (-1)^{n-r} \sum_{J\in\binom{N}{r}} B_{JJ} = (-1)^{n-r}\mathrm{tr}(\Lambda^r(B)),$$

where tr denotes the trace (Greub [78], Prop.7.5.1), and thus

$$(-1)^{n-r}c_{n-r}(AB) = \mathrm{tr}(\Lambda^r(AB)) = \mathrm{tr}(\Lambda^r(A)\Lambda^r(B))$$
$$= \sum_{I,J\in\binom{N}{r}} A_{JI}B_{IJ},$$

where $A\in M_n(F)$ is our given matrix. Now let $r = \mathrm{rank}(A)$, and consider the polynomial

$$h_A = \sum_{I,J\in\binom{N}{r}} A_{JI}Y_{IJ} \in F[y_{11},y_{12},\ldots,y_{nn}]$$

where Y is the n×n-matrix whose (i,j)-entry is the indeterminate y_{ij}. Since some A_{JI} is nonzero and no cancellation occurs, h_A is nonzero. For any $B\in M_n(F)$ we have

$$h_A(B) \neq 0 \Rightarrow c_{n-r}(AB) \neq 0$$
$$\Rightarrow r \le \mathrm{rank}_{alg}(AB) \le \mathrm{rank}(AB) \le \mathrm{rank}(A) = r$$
$$\Rightarrow \mathrm{rank}_{alg}(AB) = r,$$

and it is sufficient to show that

$$\#\{B\in M_n(P) : h_A(B)=0\} \le \tfrac{3}{4}p^{n^2}.$$

In the case $p \geq \frac{4}{3}n$, it follows from Lemma 1 below that

$$\#\{B \in M_n(P):\ h_A(B)=0\} \leq \frac{n}{p}\, p^{n^2} \leq \frac{3}{4}\, p^{n^2}.$$

Now assume that F is finite and P=F. Let

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

with rank(D) = r. Then $h_D = Y_{\{1,\ldots,r\},\{1,\ldots,r\}}$, and by the Lemma 2(ii) below

$$\#\{B \in M_n(F):\ h_D(B)=0\} \leq \frac{3}{4} p^{n^2}.$$

Now there exist $U,V \in GL_n(F)$ such that $A = UDV$. For any $B \in M_n(F)$ we have

$$\begin{aligned} h_A(B) &= tr(\Lambda^r(AB)) = tr(\Lambda^r(UDVB)), \\ h_D(B) &= tr(\Lambda^r(DB)) = tr(\Lambda^r(U)\Lambda^r(DB)\Lambda^r(U)^{-1}) \\ &= tr(\Lambda^r(UDBU^{-1})). \end{aligned}$$

We get the desired estimate, which completes the proof of Theorem 3:

$$\begin{aligned} &\#\{B \in M_n(F):\ h_A(B)=0\} \\ &= \#\{B \in M_n(F):\ tr(\Lambda^r(UDVB))=0\} \\ &= \#\{B \in M_n(F):\ tr(\Lambda^r(UDBU^{-1}))=0\} \\ &= \#\{B \in M_n(F):\ h_D(B)=0\} \leq \frac{3}{4} r^{n^2}. \end{aligned}$$

The second equality follows from the fact that $B \mapsto V^{-1}BU^{-1}$ is a bijection from $M_n(F)$ to $M_n(F)$. $\square$

For the two lemmas that establish the probabilistic estimates, we assume the following set-up which is slightly more general than required: A field F, $P \subseteq F$ finite with $p \geq 2$ elements, $d, n \in \mathbb{N}$, indeterminates $x_{11}, x_{12}, \ldots, x_{nn}$ over F, and an $n \times n$-matrix $G = (g_{ij})$ of univariate polynomials $g_{ij} \in F[x_{ij}]$ of degree $\leq d$. In Lemma 1, we estimate the probability that a linear combination of minors of G is zero when the entries are randomly chosen from P.

Lemma 1 Let $f = \sum c_{IJ} G_{IJ} \neq 0$, where $c_{ij} \in F$ and the sum is over all $I, J \subseteq \{1, \ldots, n\}$ with $\#I = \#J$. Then the probability that f vanishes is $\leq nd/p$.

Proof Let $q = p^{-n^2} \#\{A \in M_n(P) \colon f(A) = 0\}$ be the probability that f vanishes. We show that $q \leq nd/p$ by induction on n. The case $n=1$ being obvious, we can assume that $n \geq 2$, f is non-constant, and that x_{nn} occurs in f. Thus

$$f = g_{nn} h + a,$$

$$h = \sum_{\substack{(I,J) \\ n \in I \cap J}} c_{IJ} G_{I \setminus \{n\}, J \setminus \{n\}} \neq 0,$$

$$g_{nn} \in F[x_{nn}] \setminus F,$$

and x_{nn} does not occur in a. Then h satisfies the hypothesis for Lemma 1 with n replaced by $n-1$, and

$$\begin{aligned} &\#\{A \in M_n(P) \colon f(A) = 0\} \\ \leq\ &\#\{A \in M_n(P) \colon h(A) = 0\} + \#\{A \in M_n(P) \colon \\ &\quad h(A) \neq 0 \text{ and } g_{nn}(A) = -a(A) h(A)^{-1}\} \\ \leq\ &(n-1) d\, p^{n^2-1} + d p^{n^2-1} = nd p^{n^2-1}, \end{aligned}$$

hence $q \leq nd/p$. $\square$

A similar inductive argument shows that for an arbitrary polynomial f in m indeterminates and of degree $\le d$ in each indeterminate the probability of vanishing is $\le md/p$. Applying this general observation to the above situation would yield the estimate $q \le n^2 d/p$.

Lemma 2 gives a sharp estimate of the probability that a matrix of polynomials is singular. We introduce the function $u(t,n) = 1 - \prod_{1\le i\le n} (1-t^i)$ for $0\le t\le 1$ and $n \in \mathbb{N}$. u is monotonely increasing in both arguments.

<u>Lemma 2</u> In the set-up described before Lemma 1, assume that no g_{ij} is constant, and let q be the probability that det(G) is zero. Then

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

(ii) if $d=1$, then $q < 3/4$.

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

<u>Proof</u> Let $t = d/p$.

(i) We have to show that

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

where g(A) is the matrix with (i,j)-entry $g_{ij}(A_{ij})$. For $1\le r\le n$, 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 $I \subseteq \{1,\dots,n\}$ such that $\#I = r-1$ and $A_{\{1,\dots,r-1\},I} \ne 0$. 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 $i \in I$. Hence there are at most $p^{r-1}d^{n-r+1}$ vectors $z \in T_1$ such that g(z) is linearly dependent on the rows of g(A). Thus

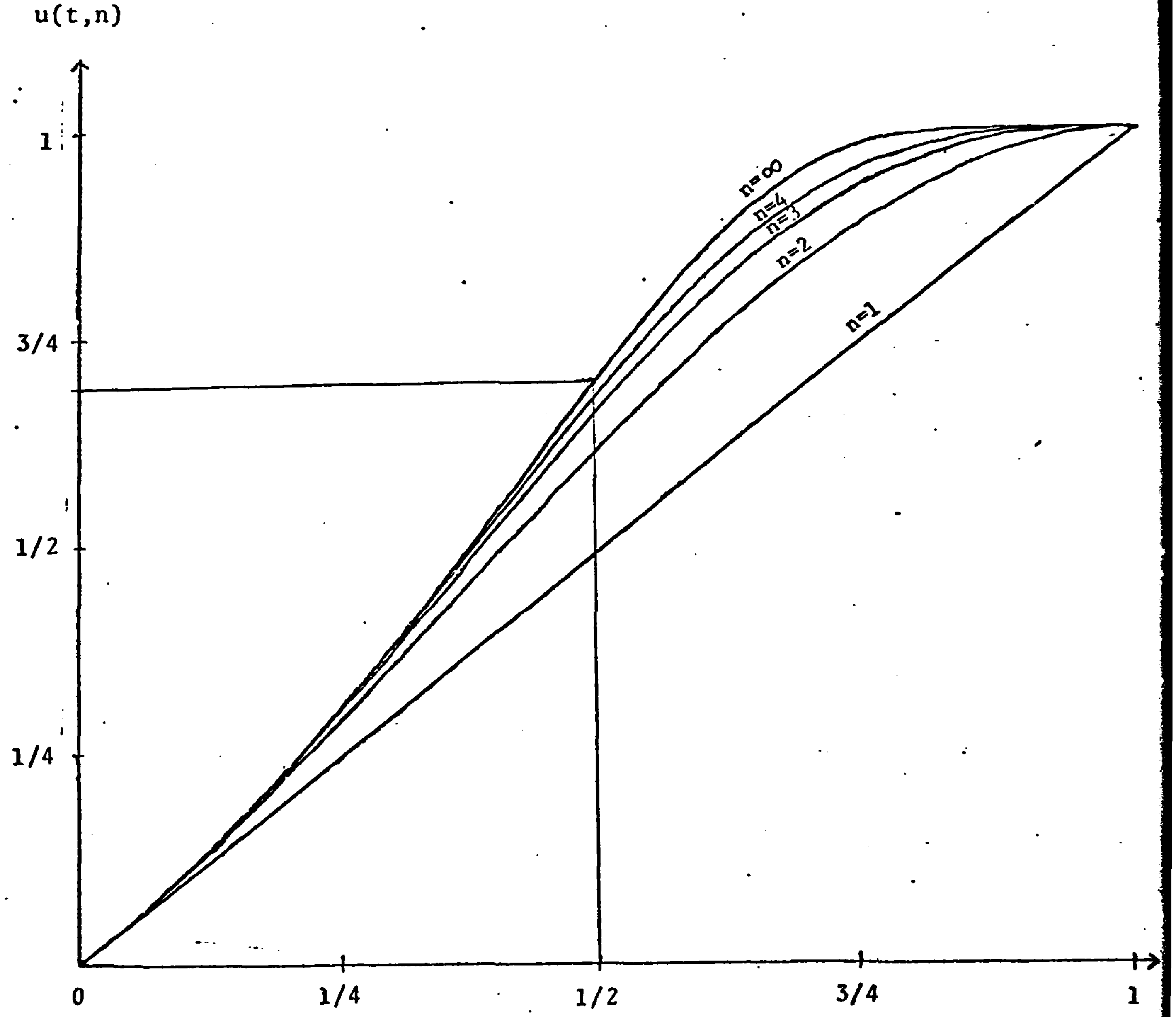

Figure. The upper bound u(t,n) on the probability that an n×n-matrix is singular. The entries of the matrix are non-constant polynomials of degree $\leq d$ over an arbitrary field F whose arguments are randomly chosen from a subset of F with p elements, and $t=d/p$. The values $n=1,2,3,4$ are shown,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

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

and by induction on r we get

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

Thus

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

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

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

The product in v is uniformly convergent on every interval $[0,a]$ with $a<1$ (see Henrici [77], §8.2 for a discussion and the relation of v with combinatorial questions), and v is a monotonely increasing function. In particular, for $0\leq t\leq 1/2$ and $n\in\mathbb{N}$ we have

$$0 \leq u(t,n) < v(t) \leq v(1/2) = 0.7112\ldots < 3/4.$$

This proves (ii), since then $t = 1/p \leq 1/2$.

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

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

follows. This implies $q = u(t,n)$. □

Statement (iii) shows that the estimate in (i) is sharp. The Figure shows the increasing functions $u(t,n)$ for $1\leq n\leq 4$, and their limiting curve $v(t) = u(t,\infty)$. We remark that the estimate of (i) also holds when the determinant is replaced by the permanent; the proof is slightly different.

When interpreted as the decision problem "is A in $U_r = \{B \in M_n(F)\colon \text{rank}(B) \leq r\}$?", the above Monte Carlo algorithm always answers correctly if $A \in U_r$, and correctly with probability $\geq 1/4$ if $A \notin U_r$. We can improve this behaviour to get an algorithm that always answers correctly and with high probability has a low parallel running time.

<u>Theorem 4</u>. For any field F, there exists a Monte Carlo 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 branch and that either returns the rank of the input matrix (and a maximal nonsingular minor) or reports that it failed. The probability of the second case is $\le 2^{-n}$.

<u>Proof.</u> Let $A\in M_n(F)$. For $1\le i\le n$ let $A_i\in M_n(F)$ consist of the first i rows of A and zero rows otherwise, and $r_i = \mathrm{rank}(A_i)$. Apply the algorithm of Theorem 3 independently 6n times in parallel to each A_i, and let s_i be the maximum of the numbers computed for A_i. Thus $s_i\le r_i$, and for each i

$$\mathrm{Prob}(\{s_i<r_i\}) \le (3/4)^{6n}.$$

Let $I = \{i: 1\le i\le n \text{ and } s_{i-1}<s_i\}$ (with $s_0=0$), and $A'\in M_n(F)$ be the matrix whose i-th row is the i-th 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 $J \subseteq \{1,\dots,n\}$. The parallel time for all this is $O(\log^2 n)$, and

$$\mathrm{Prob}(\{\text{the } I\times J\text{-minor of } A \text{ is not a maximal (square) nonsingular minor}\}) \le 2n(\tfrac{3}{4})^{6n} \le 2^{-n}.$$

If $\#I \ne \#J$ 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 $O(\log^2 n)$. If either $A_{IJ}=0$ or some such A_{KL} is nonzero, then report failure. Otherwise return $s_n = \mathrm{rank}(A)$ and the $I\times J$-minor as a maximal nonsingular minor. $\square$

It is now clear what is an appropriate collection (f,S) for RANK(n). For $0\le r\le n$, we have

$$S_r = \{A\in F^{n^2} : \operatorname{rank}(A)=r\}$$

and $f: S_r \to F^{r^2}$ maps A to its "top left" nonsingular $r\times r$-minor which is defined along the lines of the above proof (identifying F^{n^2} and $M_n(F)$ in a natural way).

Thus the condition is that $A_{IJ} \ne 0$, for any $k<i\in I$ the rows with index $(I\setminus\{i\})\cup\{k\}$ of A have rank $<r$, and for any $k<j\in J$ the columns with index $(J\setminus\{j\})\cup\{k\}$ of the I-rows of A have rank $<r$. If one wants f to consist of rational functions on each set of the partition, one has to further split up S_r according to the value of (I,J).

The rank algorithm over $\mathbb{R}$ of Ibarra-Moran-Rosier can be interpreted as avoiding the Monte Carlo aspect by noting that for $B=A^t$

$$h_A(B) = \sum_{I,J\in\binom{N}{r}} A_{IJ}^2 \ne 0.$$

Using their algorithm 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 a similar improvement for arbitrary fields? Of course our Monte Carlo algorithm yields non-uniform 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?

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 [82] and Valiant [82]). It is natural then to study the relative complexity of these problems.

Valiant [82] introduced the notion of algebraic projections as a strong type of reducibility between polynomials. For collections $P = (P(n))$ 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 transformations. 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 collections 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 [76] has the following reductions:

a) Over any field,

INVERSION $\approx$ NONSINGULAR EQUATIONS

$\leq$ DETERMINANT $\leq$ 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

CHARACTERISTIC POLYNOMIAL $\lesssim$ DETERMINANT.

This holds non-uniformly over arbitrary fields, by extending the field by the necessary roots of unity.

c) If the field has characteristic zero, then

DETERMINANT $\lesssim$ 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).

For any field, we have the following reductions:

a) BASIS $\approx$ RANK $\lesssim$ NULLSPACE $\lesssim$ RANK + INVERSION,

b) EQUATIONS $\lesssim$ RANK + INVERSION.

<u>Proof</u> a) 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 $\mathrm{rank}(v_1,\ldots,v_{i-1}) < \mathrm{rank}(v_1,\ldots,v_i)$.

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 $r+1\le i\le n$, we can solve the nonsingular systems $Mx_i = y_i$, when y_i denotes the first r rows of the i-th column of A. A basis for the nullspace of A then consists of the vectors $\begin{pmatrix} x_i \\ 0 \\ \vdots \\ -1 \\ \vdots \\ 0 \end{pmatrix}$, $r<i\le n$, that is, x_i is extended by 0's except for a -1 in the i-th position.

b) 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 non-uniform 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 non-uniform 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 [82]).

6. Conclusion

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 Monte Carlo 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 Berlekamp's [70] factoring algorithm), continued fraction 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 [82]).

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.

Acknowledgement

We are very grateful to Steve Cook for an observation which led to Theorem 2, and to both Steve Cook and Les Valiant for pointing out relations with combinatorial problems. Thanks also go to Yaacov Yesha for pointing out one of the references, and to Alain Fournier for furnishing the graph of the Figure.

References

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

W. Baur, V. Strassen, The computational complexity of partial derivatives, to appear in Theor. Comp. Science.

E.R. Berlekamp, Factoring polynomials over large finite fields, Math. Comp 24 (1970), 713-735.

A. Borodin, Structured vs. general models in computational complexity, in: Logic and Algorithmic, Symposium in honour of Ernst Specker, Monographie No. 30 de l'Enseignement Mathématique, 1982, 47-65.

D. Coppersmith and S. Winograd, On the asymptotic complexity of matrix multiplication, submitted to SIAM J. Computing.

L. Csanky, Fast parallel matrix inversion algorithms, SIAM J. Computing 5(1976), 618-623.

J. Edmonds, Systems of distinct representatives and linear algebra, J. Res. National Bureau of Standards, 71B, 4(1967), 241-245.

T. Feather, private communication, November 1981.

S. Fortune and J. Wyllie, Parallelism in random access machines, Proc. 10th ACM Symposium on Theory of Computing, San Diego, California 1978, 114-118.

J. von zur Gathen, Fast parallel algorithms for algebraic problems, to appear.

L.M. Goldschlager, R.A. Shaw and J. Staples, The maximum flow problem is log space complete for P, Preprint 1981.

W. Greub, Multilinear algebra, Springer Verlag, 2nd ed. 1978.

P. Henrici, Applied and computational complex analysis, Vol. 2, John Wiley & Sons, 1977

O. Ibarra, S. Moran and L.E. Rosier, A note on the parallel complexity of computing the rank of order n matrices, Information Processing Letters 11(1980), 162..

N. Pippenger, On simultaneous resource bounds, Proc. IEEE 20th Annual FOCS, October 1979.

V. Strassen, Vermeidung von Divisionen, J. Reine Angew. Math. 264(1973), 184-202.

V. Strassen, The computational complexity of continued fractions, Proc. 1981 ACM Symp. on Symbolic and Algebraic Computation, Snowbird, Utah 1981, 51-67.

L.G. Valiant, S. Skyum, S. Berkowitz and C. Rackoff, Fast parallel computation of polynomials using few processors, Preprint 1981.

L. Valiant, Reducibility by algebraic projections, in: Logic and Algorithmic, Symposium in honour of Ernst Specker, Monographie No. 30 de l'Enseignement Mathématique, 1982, 365-380.

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Computer Science Department, Cornell University, Ithaca, New York 14853	2a. REPORT SECURITY CLASSIFICATION
	2b. GROUP

3. REPORT TITLE

FAST PARALLEL MATRIX AND GCD COMPUTATIONS

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Technical Report

5. AUTHOR(S) *(First name, middle initial, last name)*

Allan Borodin, Joachim von zur Gathen, John E. Hopcroft

6. REPORT DATE April 1982	7a. TOTAL NO. OF PAGES 27	7b. NO. OF REFS 19

8a. CONTRACT OR GRANT NO. N00014-76-C-0018	9a. ORIGINATOR'S REPORT NUMBER(S) TR 82-487
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	

10. DISTRIBUTION STATEMENT
Distribution of manuscript is unlimited.

[illegible]

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY

13. ABSTRACT

We present parallel algorithms to compute the determinant and characteristic polynomial of n×n-matrices and the gcd of polynomials of degree $\le n$. The algorithms use parallel time $O(\log^2 n)$ and a polynomial number of processors. We also give a fast parallel Monte Carlo algorithm for the rank of matrices. All algorithms work over arbitrary fields.

DD FORM 1 NOV 65 1473 (PAGE 1)
S/N 0101-807-6801

UNCLASSIFIED
Security Classification

UNCLASSIFIED
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
computational complexity						
parallel computation						
rank						
greatest common divisor (gcd)						

DD FORM 1 NOV 65 1473 (BACK)
(PAGE 2)

UNCLASSIFIED
Security Classification