COMBINATORICA 7 (1) (1987) 101—104

A FAST PARALLEL ALGORITHM TO COMPUTE THE RANK OF A MATRIX OVER AN ARBITRARY FIELD

K. MULMULEY

Received 12 January 1986

It is shown that the rank of a matrix over an arbitrary field can be computed in $O(\log^2 n)$ time using a polynomial number of processors.

1. Introduction

Solving a system of linear equations is undoubtedly the most basic problem in linear algebra, hence it is of interest to know its parallel complexity. We already know this complexity when the system has full rank; this was done for the characteristic zero case in [4] and for an arbitrary characteristic in [1] and [3]. In general one needs to know the rank of the system to be able to solve it. It was shown in [6] that computing the rank of a matrix over the real or complex field is in NC^2. For an arbitrary field a randomized $O(\log^2 n)$ time algorithm, which used a polynomial number of processors, was given in [3]. (Here the basic unit of time is one field operation.) In this paper we shall give a *deterministic* $O(\log^2 n)$ time algorithm, which uses a polynomial number of processors, to compute the rank of a matrix over an arbitrary field.

The applications of this result will be many. Many problems which hitherto had only randomized polylog time parallel algorithms will now have deterministic polylog time parallel algorithms. These include many problems in linear algebra, notably solving a system of linear equations over an arbitrary field (see [3]), group theoretic problems like finding the order, the derived series, a composition series, testing membership if the permutation group is solvable, finding the center, a central composition series, and pointwise stabilizers of sets, if the permutation group is nilpotent (see [7], [8]), factoring polynomials over finite fields of small characteristic (the randomized algorithm is essentially the one of Berlekamp). In addition finding the gcd of many polynomials and the squarefree decomposition of polynomials are now in NC^2 instead of NC^3 [5].

Also appeared in ACM Symposium on Theory of Computing, May 28—30, 1986 Berkeley, California. Research supported by Miller Fellowship, University of California, Berkeley.

AMS subject classification (1980): 68 C 05

2. The algorithm

Suppose we are given an $n \times n$ matrix A over an arbitrary field F and we want to calculate its rank. Let $P(t)$ be its characteristic polynomial and $K(A) = \bigcup_{l \geq 1} \text{Kernel}(A^l)$ be its generalized eigenspace corresponding the eigenvalue zero. Let m be the highest integer such that t^m divides $P(t)$. Then $m = \dim(K(A))$; this is best seen by first taking the algebraic closure of the base field (this leaves $\dim(K(A))$ and the characteristic polynomial invariant) and then by looking at the Jordan canonical form of A. If $K(A) = \text{Kernel}(A)$ then $m = \dim(\text{Kernel}(A)) = \text{nullity}(A)$, and hence the rank of A can be calculated by simply computing the characteristic polynomial $P(t)$. Of course, $K(A)$ need not be equal to $\text{Kernel}(A)$, but we shall soon see a transformation which achieves this.

First, we can assume that A is square and symmetric, because if it is not, one can consider instead the matrix

$$\begin{bmatrix} 0 & A \\ A^t & 0 \end{bmatrix}$$

which has twice the rank of the original matrix. Secondly, one notes that the rank remains invariant under an extension of the base field. Thus one is at liberty to extend the field as long as the computation in the new field does not become prohibitively expensive. Let us extend our field by adding one transcendental element x to obtain $G = F(x)$, the field of rational functions in one variable. Let us construct the matrix $C = XA$ over the field G, where X is a diagonal matrix with $X_{ii} = x^{i-1}$, $(1 \leq i \leq n)$. As X is nonsingular, it is clear that the rank remains invariant under this transformation, i.e. $\text{rk}(C) = \text{rk}(A)$. Moreover,

Lemma 1. $\text{rk}(CC) = \text{rk}(C)$.

Proof. It is sufficient to prove that $\text{rk}(AXA) = \text{rk}(A)$ because then

$$\text{rk}(CC) = \text{rk}(XAXA) = \text{rk}(AXA) = \text{rk}(A) = \text{rk}(C).$$

Obviously $\text{rk}(AXA) \leq \text{rk}(A)$. Thus we only have to prove the other inequality.

Suppose $AXAu(x) = 0$, where $u(x)$ is a vector whose entries can be assumed to be polynomials in x without any loss of generality. Suppose to the contrary that $v(x) = Au(x) \neq 0$. Let $v(z)$ be a vector obtained by substituting a new transcendental z for x. We get

$$\begin{aligned} \sum_{i=1}^{n} v_i(z) v_i(x) x^{i-1} &= v^t(z) X v(x) \\ &= u^t(z) A^t X A u(x) \\ &= u^t(z) A X A u(x) \\ &= 0. \end{aligned}$$

Denote by m_i the degree of $v_i(x)$, the ith component of $v(x)$, and let $m = \max\{m_i\}$. Let k be the maximum integer such that $m_k = m$. Then it is clear that

the term corresponding to the monomial $z^{m_k}x^{m_k}x^{k-1}=z^{m_k}x^{m_k+k-1}$ in the expansion of

$$\sum_{i=1}^{n} v_i(z)v_i(x)x^{i-1}$$

can not be cancelled, and hence the sum is nonzero, a contradiction. ∎

Considering C as a linear endomorphism of $V=G^n$ the lemma implies that $\text{Kernel}(C)\cap C(V)=\{0\}$. $(C(V)=\{C(v)|v\in V\})$. Hence V is a direct sum: $V=\text{Kernel}(C)\oplus C(V)$ and the restriction of C is an automorphism of $C(V)$. This means $\text{Kernel}(C^l)=\text{Kernel}(C)$ for all $l\geqq 1$, and $K(C)=\bigcup_{l\geqq 1}\text{Kernel}(C^l)=\text{Kernel}(C)$. As noted before, this implies that $m=\dim(\text{Kernel}(C))$, where m is the highest integer such that t^m divides the characteristic polynomial $Q(t)$ of C. The rank of C, and hence that of A, can now be calculated by computing $Q(t)$, which can be done in $O(\log^2 n)$ time with $O(n^{4.5})$ processors using the algorithm in [2].

Though the knowledge of the canonical form makes it easy to see what is going on, it is easy to see that m equals the nullity of C without this knowledge. For $V=\text{Kernel}(C)\oplus C(V)$ implies that in an appropriate basis C assumes the block form

$$\begin{bmatrix} B & 0 \\ 0 & 0 \end{bmatrix}$$

where B is nonsingular, from which the result follows.

Thus the following is a simple algorithm to find the rank of a symmetric $n\times n$ matrix A over an arbitrary field F. (We have already seen a simple transformation to use if A is not symmetric and square.) Let X be a diagonal matrix in an indeterminate x such that $X_{ii}=x^{i-1}$, $(1\leqq i\leqq n)$.

1. compute $Q(t)=\det(tI-XA)$.

2. return $n-m$, where m is the highest degree such that t^m divides $Q(t)$.

The algorithm takes $O(\log^2 n)$ time using $O(n^{4.5})$ processors.

It is also interesting to consider a randomized version of the above algorithm. We can eliminate the indeterminate x and thus gain in speed, by choosing for it a random value from the field F or its suitable algebraic extension. Note that this randomized algorithm uses only one random element, whereas the algorithm in [3] uses two random matrices. Moreover, if the original matrix A was a sparse or band matrix, this property will be preserved in the transformed matrix.

3. Acknowledgement

I am grateful to L. Babai, Dick Karp, H. W. Lenstra, Gary Miller, Michael Rabin and Avi Widgerson for useful discussions and comments. The proof of Lemma 1 is a simplification over my original proof due to H. W. Lenstra.

References

[1] S. BERKOWITZ, On computing the determinant in small parallel time using a small number of processors, *Inform. Process. Lett.*, **18** (1984), 147—150.
[2] A. BORODIN, S. A. COOK and N. PIPPENGER, Parallel computation for well-endowed rings and space bounded probabilistic machines, *Information and Control* **58** (1983), 113—136.
[3] A. BORODIN, J. VON ZUR GATHEN and J. HOPCROFT, Fast parallel matrix and GCD computations, *Information and Control* **52** (1982), 241—256.
[4] L. CSÁNKY, Fast parallel matrix inversion algorithms, *SIAM J. Comput.*, **5** (1976).
[5] J. VON ZUR GATHEN, private communication.
[6] 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.
[7] E. M. LUKS and P. MCKENZIE, Fast parallel computation with permutation groups, *Proc.* **25***th FOCS*, 1985, 505—514.
[8] P. MCKENZIE and S. A. COOK, The parallel complexity of the abelian permutation group membership, *Proc.* **24***th FOCS*, 1983, 154—161.

Ketan Mulmuley

EECS Department, Computer Sci. Division
University of California
Berkeley, CA 94720, U. S. A.