COMBINATORICA 7 (1) (1987) 105—113

MATCHING IS AS EASY AS MATRIX INVERSION

KETAN MULMULEY[1], UMESH V. VAZIRANI[2] and VIJAY V. VAZIRANI

Received 12 May 1986
Revised 30 July 1986

We present a new algorithm for finding a maximum matching in a general graph. The special feature of our algorithm is that its only computationally non-trivial step is the inversion of a single integer matrix. Since this step can be parallelized, we get a simple parallel (RNC^2) algorithm. At the heart of our algorithm lies a probabilistic lemma, the isolating lemma. We show other applications of this lemma to parallel computation and randomized reductions.

1. Introduction

We present a new algorithm for finding a maximum matching in a general graph. The special feature of our algorithm is that its only computationally non-trivial step is the inversion of a *single* integer matrix. Since this step can be parallelized, we get a simple parallel (RNC^2) algorithm. Because of this simplicity, the sequential version of our algorithm has some merits over the conventional matching algorithms as well.

This algorithm was obtained while solving the matching problem from the viewpoint of parallel computation. The main difficulty here is that the graph may contain exponentially many maximum matchings; how do we coordinate the processors so they seek the same matching in parallel? The key to achieving this coordination is a probabilistic lemma, the isolating lemma, which lies at the heart of our algorithm; it helps to single out one matching in the graph.

In this general form the isolating lemma applies to an arbitrary set system. This yields a relationship between the parallel complexity of an arbitrary search problem and the corresponding weighted decision problem. As an application, we give an RNC^2 algorithm for the Exact Matching problem in general graphs It is interesting to note that this problem is not known to be in P. The isolating lemma also yields a simple proof for the theorem in [19], showing the NP-hardness, under randomized reductions of instances of SAT having unique solutions.

Karp, Upfal and Wigderson first showed that matching is in Random $NC(RNC^3)$. Their algorithm also uses matrix operations, and in fact these are some of the most powerful tools for obtaining fast parallel algorithms. Several problems are known to be NC^1 reducible to computing the determinant of an integer matrix. Cook [1] defines DET to be the class of all such problems. $DET \subseteq NC^2$, and it is

---

[1] Miller Fellow, University of California, Berkeley.
[2] Work done while visiting MSRI, Berkeley, in Fall 1985.
Supported by NSF Grant BCR 85—03611 and an IBM Faculty Development Award.
AMS subject classification (1980): 68 E 10, 05 C 25

not known whether this inclusion is proper. Cook also observes that all problems known to be in NC^2 are either in AC^1 or in *DET*. One important consequence of our algorithm is that it puts the matching problem in *RDET*.

2. History

The maximum matching problem is a natural and simply-stated problem and is used as a subroutine in several computational problems. More significantly, the study of this problem has led to conceptual breakthroughs in the field of algorithms. In fact, the characterization of 'tractable problems' as 'polynomial time solvable problems' was first proposed by Edmonds [3] in the context of the general graph matching problem. Solving this problem from the viewpoint of parallel computation has also been quite fruitful.

Whereas sequential algorithms for the maximum matching problem are based on finding 'blossoms' and 'augmenting paths' in graphs (see [3]), the known parallel algorithms require a new approach; they use *probabilistic and algebraic methods*. In fact, the matching problem emerged from algebra around the turn of this century in the works of Petersen, Frobenius and König (for a detailed history see [12]). A key ingredient in the new approach is a theorem proved by Tutte in 1947 [18], based on the work of Pfaff on skew-symmetric matrices. It states that a graph has a perfect matching iff a certain matrix of indeterminates, called the Tutte matrix, is non-singular. Motivated by an algorithmic use of this theorem, Edmonds [4] studied the complexity of computing determinants. He gave a modified Gaussian elimination procedure for computing the determinant of an integer matrix in a polynomial number of bit operations, and stated the open problem of efficiently deciding whether a matrix of indeterminates is non-singular.

The first algorithm based on Tutte's theorem was given by Lovász [10]. Using the fundamental insight that polynomial identities can be efficiently tested by randomization (see [17]), Lovász reduced, the decision problem, 'Does the given graph have a perfect matching?' to testing if a given integer matrix is non-singular. Since the latter problem is in NC^2 (see Csánky [2]), this yields a (Monte Carlo) RNC^2 algorithm for the former problem. Rabin and Vazirani [16] extended this approach, using a theorem of Frobenius, to give a simple randomizing algorithm which finds a perfect matching by sequentially inverting $|V|/2$ matrices.

The search problem, i.e. actually finding a perfect matching in parallel, is much harder. The first parallel (RNC^3) algorithm for this long-standing open problem was given by Karp, Upfal and Wigderson [7]. They use the Tutte matrix to implement (in RNC^2) a 'rank' function. Using this, their algorithm probabilistically prunes out edges from the graph in $O(\log |V|)$ stages; finally it is left with a perfect matching. This algorithm is also Monte Carlo in that it may fail to give a perfect matching. Using the Gallai—Edmonds Structure Theorem (see [11]) Karloff [6] gives a complementary Monte Carlo (RNC^2) algorithm for bounding the size of a maximum matching from above, thus yielding a Las Vegas extension.

Our algorithm is conceptually different in that it directly finds a perfect matching in the graph. It is somewhat faster (RNC^2) and requires $O(n^{3.5}m)$ processors.

3. The isolating lemma

Definition. A *set system* (S, F) consists of a finite set S of elements, $S=\{x_1, x_2, \ldots, x_n\}$, and a family F of subsets of S, i.e. $F=\{S_1, S_2, \ldots S_k\}$, $S_j \subseteq S$, for $1 \leqq j \leqq k$.

Let us assign a weight w_i to each element $x_i \in S$ and let us define the weight of the set S_j to be $\sum_{x_i \in S_j} w_j$.

Lemma 1. *Let (S, F) be a set system whose elements are assigned integer weights chosen uniformly and independently from $[1, 2n]$. Then,*

$$\Pr\ [\textit{There is a unique minimum weight set in } F] \geqq \frac{1}{2}.$$

Proof. Fix the weights of all elements except x_i. Define the *threshold* for element x_i, to be the real number α_i such that if $w_i \leqq \alpha_i$ then x_i is contained in some minimum weight subset, S_j, and if $w_i > \alpha_i$ then x_i is in no minimum weight subset.

Clearly, if $w_i < \alpha_i$, then the element x_i must be in *every* minimum weight subset. Thus ambiguity about element x_i occurs iff $w_i = \alpha_i$; since in this case there is a minimum weight subset that contains x_i and another which does not. In this case we shall say that the element x_i is *ambiguous*.

We now make the crucial observation that the threshold, α_i, was defined without reference to the weight, w_i, of x_i. It follows that α_i is *independent* of w_i. Since w_i is a uniformly distributed integer in $[1, 2n]$,

$$\Pr\ [\textit{Element } x_i \textit{ is ambiguous, i.e. } w_i = \alpha_i] \leqq \frac{1}{2n}.$$

Since S contains n elements,

$$\Pr\ [\textit{There exists an ambiguous element}] \leqq \frac{1}{2n} \times n = \frac{1}{2}.$$

Thus, with probability at least 1/2, no element is ambiguous. In this case each element is either in every minimum weight subset or in none. It follows that the minimum weight subset is unique. ∎

Notice that by the same argument, the maximum weight set will be unique with probability at least 1/2 as well.

4. The matching algorithm

We will first present a parallel algorithm for the following problem:

Input: Graph $G(V, E)$, having a perfect matching.

Problem: Find a perfect matching in G.

We will view the edges in E and the set of perfect matchings in G as a set system. Let us assign random integer weights to the edges of the graph, chosen uniformly and independently from $[1, 2m]$, where $m=|E|$. Now, by Lemma 1, the minimum

weight perfect matching in G will be unique with probability at least 1/2. Our parallel algorithm will pick out this perfect matching. We will first introduce Tutte's Theorem.

Notation. We will represent the $(i,j)^{\text{th}}$ element of matrix A by (lower case) a_{ij}, the minor obtained by removing the i^{th} row and the j^{th} column by A_{ij}, the determinant of A by $|A|$, and the adjoint of A by adj (A).

Definition. Given a graph $G(V, E)$, the adjacency matrix of G is an $n \times n$ symmetric matrix D such that $d_{ij}=1$ if $(v_i, v_j) \in E$, and 0 otherwise. The *Tutte matrix* of G is an $n \times n$ skew-symmetric matrix A, obtained as follows from D: if $d_{ij}=d_{ji}=1$, replace them by indeterminates x_{ij} and $-x_{ij}$, so that the entries above the diagonal are positive, and leave the 0 entries of D unchanged.

Theorem (Tutte [18]). *Let $G(V, E)$ be a graph and let A be its Tutte matrix. Then $|A| \not\equiv 0$ iff there is a perfect matching in G.*

We will obtain an integer matrix B from the Tutte matrix by substituting for the indeterminates x_{ij} the integers $2^{w_{ij}}$, where w_{ij} is the weight assigned to the edge (v_i, v_j).

Lemma 2. *Let $G(V, E)$ be a graph with weights assigned to its edges, and let B be the matrix described above. Suppose the minimum weight perfect matching in G is unique, and its weight is w. Then $|B| \neq 0$; moreover, the highest power of 2 which divides $|B|$ is 2^{2w}.*

Proof. The proof is an extension of the proof of Tutte's Theorem. For each permutation σ on $\{1, 2, \ldots, n\}$, define

$$value\ (\sigma) = \prod_{i=1}^{n} b_{i\sigma(i)}.$$

Thus $value(\sigma) \not\equiv 0$ iff $(v_i, v_{\sigma(i)}) \in E$, for $1 \leqq i \leqq n$. In this case say that σ is *non-vanishing*.

By definition, $|B| = \sum_{\sigma} sign(\sigma) \times value(\sigma)$

where $sign(\sigma)$ is $+1$ if σ is an even permutation, and -1 otherwise.

Define the *trail* of a non-vanishing permutation σ to be the subgraph of G consisting only of the edges $(v_i, v_{\sigma(i)})$, for $1 \leqq i \leqq n$. Clearly, each vertex will have two such edges (may be with repetition) incident at it, and the subgraph will in general consist of disjoint cycles and single edges traversed twice. Corresponding to each perfect matching M in G there is a non-vanishing permutation whose trail is M itself. For every permutation σ containing an odd cycle, there is a corresponding permutation σ' which differs from σ only in that this odd cycle is traversed in the reverse direction. Clearly, $value(\sigma) = -value(\sigma')$ and $sign(\sigma) = sign(\sigma')$. Therefore, the permutations containing odd cycles cancel each other out, and do not contribute to $|B|$.

Let M be the minimum weight perfect matching, and let w be its weight. The value of the permutation whose trail is M is $(-1)^{n/2} 2^{2w}$. We want to argue that the value of any other non-vanishing permutation which does not contain an odd cycle must be a higher power of 2. Certainly this is true if the trail of the permutation is a perfect matching. On the other hand, the edges of an even cycle can be partitioned

into two matchings. Therefore, if the trail of a permutation contains even cycles, we can demonstrate two perfect matchings M_1 and M_2 whose union is the trail of σ. Clearly $|value(\sigma)| = 2^{w(M_1)+w(M_2)} > 2^{2w}$. ∎

Thus by evaluating $|B|$, we can determine the weight of the minimum weight matching. The next lemma will enable us to obtain the matching itself.

Lemma 3. *Let M be the unique minimum weight matching in G, and let w be its weight. The edge (v_i, v_j) belongs to M iff*

$$\frac{|B_{ij}|\,2^{w_{ij}}}{2^{2w}} \text{ is odd.}$$

Proof. First notice that

$$|B_{ij}|\,2^{w_{ij}} = \sum_{\sigma:\,\sigma(i)=j} sign(\sigma) \times value(\sigma).$$

Since n is even, if a non-vanishing permutation contains an odd-cycle in its trail, then it contains at least two such cycles. Now by the argument in Lemma 2, such permutations cancel each other out and do not contribute to $|B_{ij}|\,2^{w_{ij}}$.

If $(v_i, v_j) \in M$, the permutation whose trail is M has value $\pm 2^{2w}$, and the value of every other permutation is a higher power of 2. On the other hand, if $(v_i, v_j) \notin M$, all permutations have values which are higher power of 2. The lemma follows. ∎

The algorithm to find M is now straightforward:
Step 1: Compute $|B|$, and obtain w.
Step 2: Compute adj(B); its $(j, i)^{\text{th}}$ entry will be the minor $|B_{ij}|$.
Step 3: For each edge (v_i, v_j) do in parallel:

Compute $\dfrac{|B_{ij}|\,2^{w_{ij}}}{2^{2w}}$;

If this quantity is odd, include (v_i, v_j) in the matching.

end;

Notice that the only non-trivial computational effort is involved in evaluating the determinant and adjoint of B. We will use Pan's [14] randomized matrix-inversion algorithm, which computes $|B|$ and adj(B) in order to compute B^{-1}. It requires $O(\log^2 n)$ time and $O(n^{3.5}m)$ processors for inverting an $n \times n$ matrix whose entries are m-bit integers. In comparison, there is a processor-efficient RNC^3 implementation of the algorithm of [7] which requires $O(n^{3.5})$ processors [5].

Theorem. *There is an* RNC^2 *parallel algorithm which finds a perfect matching in the given graph. It uses $O(n^{3.5}m)$ processors.*

Although the sequential version of our algorithm is less efficient than conventional matching algorithms (the most efficient of these takes $O(m\sqrt{n})$ steps [13]), it has the advantage of being easy to program, especially if a subroutine for matrix inversion is available. Rabin and Vazirani [16] had previously obtained a simple matching algorithm to address the issue of ease of programming. It would be informative to compare these two algorithms.

5. Parallel algorithms for related problems

A parallel algorithm for the perfect matching problem easily yields parallel algorithms for the following related problems. RNC^3 algorithms for these problems are given in [7]. Here we give RNC^2 algorithms.

a) We first address the problem of finding a *minimum weight perfect matching* in a graph $G(V, E)$, given edge-weights $w(e)$ for each edge $e \in E$ in *unary*. First notice that if the weight of each edge is scaled up by a factor of mn, the minimum weight perfect matchings will be lighter than the rest by at least mn. We can now use the isolating lemma to isolate one of these minimum weight matchings: to edge $e \in E$ assign the weight $mnw(e)+r_e$, where r_e is chosen uniformly and independently from $[1, 2m]$. The proof of Lemma 1 works in this setting as well. As such this algorithm will require $O(n^{3.5}mW)$ processors, where W is the weight of the heaviest edge. Hence if the edge-weights are in unary, this problem is in RNC^2. The parallel complexity of this problem when the edge-weights are given in binary is as yet unresolved.

b) The problem of finding a *maximum matching* in a graph can now be reduced to minimum weight perfect matching as follows: extend G into a complete graph by throwing in new edges. Assign weight 0 to each edge of G, and 1 to each of the new edges, and find a minimum weight perfect matching (for an alternative method see [16]).

c) The *vertex-weighted matching* problem is the following:

Input: Graph $G(V, E)$, and a positive weight for each vertex $v \in V$.

Problem: Find a matching in G whose vertex-weight is maximum. The vertex-weight of a matching is defined to be the sum of the weights of the vertices covered by the matching.

First notice that the desired matching will be a maximum matching. This is so because any non-maximum matching can be augmented into a maximum matching without unmatching any vertex in the process. Define $V' \subseteq V$ to be a *matching set* if V' is the set of vertices covered by a maximum matching in G. The solution now consists of finding the heaviest matching set, and a perfect matching in the subgraph induced by these vertices. Sort the vertices of G by decreasing weight. Two matching sets can be compared lexicographically in this sorted order.

Lemma 4. *The lexicographically largest matching set is the heaviest matching set.*

Proof. Let L and H be maximum matchings which give the lexicographically largest and the heaviest matching sets respectively. Suppose these matching sets are different. Let u be the first vertex in the sorted order where the two sets differ. The vertex u will be matched L but not in H. Consider the symmetric difference of L and H. This will have an alternating even length path from u to a vertex v, say. The symmetric difference of this path and H will yield a matching heavier than H, since v is lighter than u. The contradiction proves the lemma. ∎

We now use the RNC^2 algorithm of Vazirani and Vazirani [20] for obtaining the lexicographically largest matching set. Their algorithm is based on a generalization of Tutte's Theorem.

6. Other applications of the isolating lemma

a) Parallel complexity of search vs. decision problems.

For the case of sequential computation, search problems are reducible to the corresponding decision problems via self-reducibility. Can such a reduction be parallelized? Notice that the self-reduction process yields the lexicographically first solution. For several problems, such as maximal independent set and depth first search, finding such a solution is P-hard, even though efficient parallel algorithms exist for the unrestricted search problem (the parallel complexity of finding the lexicographically first perfect matching is as yet unresolved). This issue was first studied by Karp, Upfal and Wigderson [8]. Motivated from matroid theory, they give an RNC^2 procedure for the search problem, using an oracle for the 'rank' function.

Via the isolating lemma, we reduce a general search problem to the *weighted* decision problem, where the weights are polynomially bounded. The general search problem is 'Given a set system (S, F) find a set in F'. We will give an RNC^1 procedure for solving this problem, given an oracle for: 'Given a set system (S, F) with polynomially bounded positive integral weights for the elements of S and an integer k, is there a set in F whose total weight is less than or equal to k?' The procedure is similar to the perfect matching algorithm of Section 3. The weight of the minimum weight set is determined by binary search on k, using $O(\log n)$ calls to the weighted decision procedure. Its elements are identified in parallel by the following observation: an element x_i is in the minimum weight set iff upon increasing its weight by 1, the weight of the minimum weight set increases. Hence we can determine the elements of the minimum weight set in parallel.

Using this procedure we obtain an RNC^2 algorithm for the following problem posed by Papadimitriou and Yannakakis [15]. Interestingly enough, it is not known if this problem can be solved in (deterministic) polynomial time.

Exact Matching:

Input: A graph $G(V, E)$, a subset $E' \subseteq E$ of red edges, and a positive integer k.

Output: Find a perfect matching involving exactly k red edges.

In this case, the set system will consist of all perfect matchings which have exactly k red edges. Assume that polynomially bounded weights w_e are given to the edges $e \in E$ of G, and there is a unique minimum weight perfect matching with k red edges. The following NC^2 procedure, suggested by Lovász, will find the weight of this perfect matching: in the Tutte matrix of G, substitute 2^{w_e} for a variable x_e if $e \in E - E'$ and $2^{w_e} y$ if $e \in E'$, where y is an indeterminate. Let B be the resulting (skew-symmetric) matrix. Now, $|B| = (pf(B))^2$ where $pf(B)$ is the Pfaffian of B. Compute $pf(B)$ by computing the square-root of $|B|$ using the parallel determinant algorithm [21] followed by interpolation. The power of 2 in the coefficient of y^k will be the weight of the minimum weight perfect matching involving exactly k red edges.

b) Randomized Reductions.

We now turn to another application of the main lemma. Valiant and Vazirani [19] studied the complexity of finding solutions to instances of SAT having unique solutions. They show that this problem is NP-hard under randomized reductions.

Their proof is based on the hash-function property of $GF[2]$ inner products. the isolating lemma yields a simpler proof.

For simplicity, we consider the *CLIQUE* problem, which is parsimoniously interreducible with *SAT*. The core of the proof is illustrated by showing a randomized reduction from *CLIQUE* to *UNIQUE CLIQUE*. The *CLIQUE* problem is 'Given a graph $G(V, E)$ and an integer k, is there a clique of size k in the graph?' On the other hand, *UNIQUE CLIQUE* asks if there is exactly one clique of size k.

The reduction is as follows. First assign a random and independent weight $w(v)$ to each vertex $v \in V$, chosen from $[1, 2n]$, where $n = |V|$. By the isolating lemma, with probability at least $1/2$, the maximum weight clique will be unique in this graph. The transformed graph G' is now obtained as follows: corresponding to vertex $v \in V$, G' will have $2nk + w(v)$ vertices, with a clique on them. Corresponding to each edge (u, v) in G, each copy of u is joined to each copy of v in G'. Next choose a random integer r in $[1, 2n]$, and let $k' = 2nk^2 + kr$. The transformed problem is (G', k').

The following hold by Lemma 1:

$$(1) \qquad (G, k) \notin CLIQUE \Rightarrow (G', k') \notin UNIQUE\ CLIQUE.$$

$$(2) \qquad (G, k) \in CLIQUE \Rightarrow \Pr[(G', k') \in UNIQUE\ CLIQUE] \geq 1/4n.$$

7. Discussion

In applying the main lemma to the case of perfect matchings, it seems that substituting random integers from $[1, 2n]$ should suffice where $|V| = n$. This will improve the processor-efficiency of the parallel algorithm and the running time of the sequential Las Vegas algorithm.

An important open problem remaining is whether the maximum matching problem is in (deterministic) *NC*. Currently, incomparability graphs is the largest class of graphs for which this problem is known to be in *NC* [9]. It may be easier to solve the decision problem, 'Does the given graph have a perfect matching?', before tackling the general search problem. The following modified decision problem is known to be in *NC*, 'Does the given bipartite graph have a unique perfect matching?' [9].

Acknowledgements. We are thankful to David Aldous, Jack Edmonds, László Lovász, Éva Tardos, and Les Valiant for valuable discussions.

References

[1] S. A. Cook, A Taxonomy of Problems with Fast Parallel Algorithms, *Information and Control*, **64** (1985), 2—22.

[2] L. Csánky, Fast Parallel Matrix Inversion Algorithms. *SIAM J. Computuing*, **5** (1976), 618—623.

[3] J. Edmonds, Paths, Trees and Flowers, *Canad. J. Math.*, **17** (1965), 449—467.

[4] J. Edmonds, Systems of Distinct Representatives and Linear Algebra, *J. Res. Nat. Bureau of Standards, 71B*, **4** (1967), 241—245.

[5] Z. Galil and V. Pan, Improved Processor Bounds for Algebraic and Combinatorial Problems in RNC, *Twenty Sixth Annual IEEE Symp. on the Foundations of Computer Science*. (1985), 490—495.

[6] H. KARLOFF, A Randomized Parallel Algorithm for the Odd Set Cover Problem, *Combinatorica* **6** (1986), 387—391.
[7] R. M. KARP, E. UPFAL and A. WIGDERSON, Finding a Maximum Matching is in Random NC, *Seventeenth Annual Symp. on Theory of Computing.* (1985), 22—32.
[8] R. M. KARP, E. UPFAL and A. WIGDERSON, Are Search and Decision Problems Computationally Equivalent? *Seventeenth Annual Symp. on Theory of Computing.* (1985).
[9] D. KOZEN, U. V. VAZIRANI and V. V. VAZIRANI, NC Algorithms for Comparability Graphs, Interval graphs, and Testing for Unique Perfect Matching, *Fifth Annual Foundations of Software Technology and Theoretical Computer Science Conference* (1985), invited paper in *Theoretical Computer Science.*
[10] L. LOVÁSZ, On Determinants, Matchings and Random Algorithms, *Fundamentals of Computing Theory,* edited by L. Budach, Akademia-Verlag, Berlin, (1979).
[11] L. LOVÁSZ, *Combinatorial Problems and Exercises,* Akadémiai Kiadó, Budapest, and North-Holland, Amsterdam, (1979).
[12] L. LOVÁSZ and M. PLUMMER, *Matching Theory,* Academic Press, Budapest, Hungary, (1986).
[13] S. MICALI and V. V. VAZIRANI, An $O(\sqrt{|V|}\,|E|)$ Algorithm for Finding Maximum Matching in General Graphs, *Twenty First Annual IEEE Symp. on the Foundations of Computer Science* (1980), 17—27.
[14] V. PAN, Fast and Efficient Algorithms for the Exact Inversion of Integer Matrices, *Fifth Annual Foundations of Software Technology and Theoretical Computer Science Conference* (1985).
[15] C. H. PAPADIMITRIOU and M. YANNAKAKIS, The Complexity of Restricted Spanning Tree Problems, *Journal of the ACM,* **29** (1982), 285—309.
[16] M. O. RABIN and V. V. VAZIRANI, Maximum Matching in General Gaphs Through Randomization, submitted.
[17] J. T. SCHWARTZ, Fast Probabilistic Algorithms for Verification of Polynomial Identities, *J. of ACM,* **27** (1980), 701—717.
[18] W. T. TUTTE, The Factorization of Linear Graphs, *J. London Math. Soc.,* **22** (1947), 107—111.
[19] L. G. VALIANT and V. V. VAZIRANI, NP is as Easy as Detecting Unique Solutions, *Seventeenth Annual Symp. on Theory of Computing.* (1985), to appear in *Theoretical Computer Science.*
[20] U. V. VAZIRANI and V. V. VAZIRANI, The Two-Processor Scheduling Problem is in Random NC, *Seventeenth Annual Symp. on Theory of Computing* (1985), 11—21, submitted.
[21] A. BORODIN, S. A. COOK and N. PIPPINGER, Parallel Computation for Well-endowed Rings and Space Bounded Probabilistic *Machines, Information and Control,* **58** (1983) 113—136.

Ketan Mulmuley
Comp. Sci. Dept.
University of California
Berkeley, CA 94720
U. S. A.

Umesh V. Vazirani
MSRI, Berkeley
CA 94720
U. S. A.

Vijay V. Vazirani
Comp. Sci. Dept.
Cornell University
Ithaca, NY 14853
U. S. A.