Matching is as Easy as Matrix Inversion

Ketan Mulmuley [1]

Computer Science Department
University of California, Berkeley

Umesh V. Vazirani [2]

Aiken Computation Laboratory
Harvard University

Vijay V. Vazirani [3]

Computer Science Department
Cornell University

Abstract

A new algorithm for finding a maximum matching in a general graph is presented; its special feature being that the only computationally non-trivial step required in its execution 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 applications of this lemma to parallel computation and randomized reductions.

1. Introduction

A new algorithm for finding a maximum matching in a general graph is presented; its special feature being that the only computationally non-trivial step required in its execution 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 its general form, the isolating lemma holds for an arbitrary set system. This yields a relationship between the parallel complexity of an arbitrary search problem

[1] Miller Fellow, University of California, Berkeley.

[2] Work done during a post-doctoral position at MSRI, Berkeley.

[3] Supported by NSF Grant BCR 85-03611 and an IBM Faculty Development Award.
Currently at AT&T Bell Labs, Murray Hill, NJ 07974.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

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 result in [ValVaz] showing the NP-hardness, under randomized reductions, of instances of SAT having a unique solution.

Matching was first shown to be in *Random NC* (RNC^3) by Karp, Upfal and Wigderson [KUW1]. Their algorithm also utilizes matrix operations, and in fact these are some of the most widely used tools for obtaining fast parallel algorithms. Several problems are known to be NC^1 reducible to computing the determinant of an integer matrix. Cook [Co] defines DET to be the class of all such problems. $DET \subseteq NC^2$, and it is not known whether this inclusion is proper. All problems known to be in NC^2 are either in AC^1 or in DET [Co]. Our algorithm puts the matching problem in $RDET$.

2. History

The maximum matching problem is a natural problem, and its study 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 [Ed1] 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 [Ed1]), 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 Konig (for a detailed history see [LP]). A key ingredient in the new approach is a theorem proved by Tutte in 1947 [Tu], 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 [Ed2] 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 Lovasz [Lo1]. Using the fundamental insight that polynomial identities can be efficiently tested by randomization [Sc], Lovasz 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 [Cs], this yields a (Monte Carlo) RNC^2 algorithm for the former problem (see also [BGH]). Rabin and Vazirani [RV] 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 [KUW1]. They use the Tutte matrix to implement (in RNC^2) a 'rank' function. Their algorithm probabilistically prunes out edges from the graph; the rank function guarantees that the remaining graph has a perfect matching, and a probabilistic lemma ensures that a constant fraction of the edges are pruned at each stage. Hence, after $O(\log |V|)$ stages only a perfect matching remains. This algorithm is also Monte Carlo in that it may fail to give a perfect matching. Using the Gallai-Edmonds Structure Theorem (see [Lo2]) Karloff [Ka] 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. $F = \{S_1, S_2, \cdots S_k\}$, $S_j \subseteq S$, for $1 \le j \le k$.

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 $\sum_{x_i \in S_j} w_i$.

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

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

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 $w_i \le \alpha_i$ 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 *singular.*

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 $[1, 2n]$,

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

Since S contains n elements,

$$Pr\ [\textit{There exists a singular element}] \le \left(\frac{1}{2n}\right) \times n = 1/2.$$

Thus, with probability at least 1/2, no element is singular. The lemma follows from the observation that there is a unique minimum weight set iff no element is singular.

Notice that by the same argument, the maximum weight set will be unique with probability at least 1/2 as well. An extension of the isolating lemma is required in Theorem 2(a). In this extension we are given integers $a_1, a_2, \cdots a_k$, and the weight of set S_j is defined to be $a_j + \sum_{x_i \in S_j} w_i$, $1 \le j \le k$. The proof given above works for this case as well.

4. The Matching Algorithm

We will first consider the simpler case of a bipartite graph:

Input: A bipartite graph $G(U,V,E)$, 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.

Notation: We will represent the $(i, j)^{th}$ element of matrix A by (lower case) a_{ij}, the submatrix 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 $adj(A)$.

Let $U = \{u_1, \cdots u_n\}$, $V = \{v_1, \cdots v_n\}$, and let D be the $n \times n$ adjacency matrix of G, i.e. $d_{ij} = 1$ if $(u_i, v_j) \in E$, and 0 otherwise. Obtain an integer matrix B from D by replacing the 1's in D by $2^{w_{ij}}$, where w_{ij} is the weight assigned to the edge (u_i, v_j).

Lemma 2: Suppose the minimum weight perfect matching in $G(U,V,E)$ is unique. Let this matching be M and its weight be w. Then $|B| \neq 0$; moreover, the highest power of 2 which divides $|B|$ is 2^w.

Proof: First notice that each perfect matching in G corresponds to a permutation in S_n. For each permutation σ on $\{1, 2, \ldots, n\}$, define

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

Thus $value(\sigma) \neq 0$ iff $(u_i, v_{\sigma(i)}) \in E$, for $1 \leq i \leq n$, i.e. if σ represents a perfect matching in G. By definition,

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

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

Let σ_M be the permutation corresponding to M. Then $value(\sigma_M) = 2^w$. The value of each of the remaining permutations is either zero, or a higher power of 2. The lemma follows.

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 (u_i, v_j) belongs to M iff $\frac{|B_{ij}| 2^{w_{ij}}}{2^w}$ is odd.

Proof: First notice that

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

Let σ_M be the permutation corresponding to M. If $(u_i, v_j) \in M$, one permutation, i.e. σ_M, in the above sum will have value 2^w. The remaining permutations have value zero, or a higher power of 2. Hence $|B_{ij}| 2^{w_{ij}} / 2^w$ will be odd. On the other hand, if $(u_i, v_j) \notin M$, all permutations in the sum have value zero, or a power of 2 higher than 2^w. Hence $|B_{ij}| 2^{w_{ij}} / 2^w$ will be even. The lemma follows.

The algorithm to find M is now straightforward:

```
Procedure: Perfect Matching (G, B);

Step 1: Compute |B|, and obtain w.

Step 2: Compute adj(B); its (j, i)th entry
        will be the minor |B_ij|.

Step 3: For each edge (u_i, v_j) do in parallel:
            Compute |B_ij| 2^(w_ij) / 2^w ;
            If this quantity is odd, include
            (u_i, v_j) in the matching.
      end;
```

The algorithm for general graphs is essentially the same. The main difference is that we need to operate with the Tutte matrix of the graph.

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 [Tu]): Let $G(V, E)$ be a graph and let A be its Tutte matrix. Then $|A| \neq 0$ iff there is a perfect matching in G.

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). The algorithm given above, operated with B, gives a perfect matching in G. The proof of Lemmas 2 and 3 is more involved for general graphs, and it appears in the final paper (to appear in Combinatorica).

Notice that the only non-trivial computational effort required in the matching algorithm is the evaluation of the determinant and adjoint of B. We will use Pan's [Pa] 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 [KUW1] which requires $O(n^{3.5})$ processors [GP].

Theorem 1: There is an RNC^2 algorithm for finding a perfect matching in general graphs. The algorithm requires $O(n^{3.5}m)$ parallel 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 [MV]), it has the advantage of being easy to program, especially if a subroutine for matrix inversion is available. In [RV] a simple matching algorithm is presented, addressing 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 [KUW1]. 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 [RV]).

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 in 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 presented in [VV] for obtaining the lexicographically largest matching set. This algorithm is based on a generalization of Tutte's Theorem.

Theorem 2: The following problems are in RNC^2:
a). Finding a maximum matching in a graph.
b). Finding a minimum weight perfect matching when the edge weights are given in unary.
c). Finding a maximum vertex-weighted matching (even if the vertex weights are given in binary).

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-complete (see [Co]), even though efficient parallel algorithms exist for the unrestricted search problem (the parallel complexity of finding the lexicographically first perfect matching or the lexicographically first maximal matching is as yet unresolved). This issue was first studied by Karp, Upfal and Wigderson [KUW2]. 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.

Theorem 3: Let (S, F) be an arbitrary set system, and let O be an oracle for the weighted decision problem, 'Given polynomially bounded positive integral weights for the elements of S and a positive integer k, is there a set F whose weight is k or less?' There is an RNC^1 procedure which uses O to solve the search problem 'Find a set in F'.

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 [PY]. 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 Lovasz, 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 $|B|$ using the parallel determinant algorithm of [BCP], and then compute its square-root 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.

Theorem 4: The exact matching problem is in RNC^2.

b) Randomized Reductions

We now turn to another application of the isolating lemma. Valiant and Vazirani [ValVaz] 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 inter-reducible 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 $[1, 2nk]$, and let $k' = 2nk^2 + r$. The transformed problem is (G', k').

The following hold by Lemma 1:

(1) $(G, k) \notin$ CLIQUE $=>$ $(G', k') \notin$ UNIQUE CLIQUE.

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

7. Discussion and Future Directions

One difficulty in solving combinatorial problems in parallel is the following: on the one hand it is crucial to coordinate the processors so they seek the same solution in parallel; on the other hand, the problem of finding a solution with any special properties, such as the lexicographically first one, is typically P-complete. The isolating lemma gets around this; it induces a probability distribution on the set of matchings in the graph, and it picks one matching from this distribution. The distribution assigns a nonzero probability to each matching in the graph. Can the methods presented here be extended to achieve a uniform probability distribution, thereby obtaining *random* matching in the given graph? This will help solve a long-standing open problem, that of estimating the permanent of a 0/1 matrix. Computing the permanent exactly is $\#P$-complete [Va]; however, the problem of estimating the permanent is equivalent to the problem of generating a random perfect matching in a bipartite graph [Br], [JVV].

The main step in the matching algorithm, matrix inversion, can be accomplished in several ways: by Guassian elimination, a greedy algorithm, or by Strassen's method (see [AHU]), which is based on divide and conquer. Thus by suitably implementing matrix inversion, the matching algorithm acquires a greedy/divide-and-conquer flavour. This opens the possibility of a combinatorial algorithm for matching which has this flavour. Notice that the algorithm presented relies critically on the choice of a suitable generalization of matching: the weighted problem with the additional condition that there is a unique minimum weight perfect matching. Whereas it is unlikely that there is a greedy/divide-and-conquer algorithm for matching itself, the results presented in this paper indicate that it is not unreasonable to expect a combinatorial algorithm for the generalized problem, or a suitable modification.

In applying the isolating 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.

Notice that the proof of the isolating lemma relies on the independence of the weights of the elements. It would be interesting to study how crucial the role of independence is. The semi-random source, introduced in [SV] and [Vaz], mathematically models dependence using the notion of an adversary. Thus one could study the probability that the minimum weight set is unique, if the weights of the elements are assigned by a semi-random source. One possible formulation is the following:

Let (S,F) be a set system with $S = \{x_1, \cdots x_n\}$. Each element of S is assigned a label by the roll of an $m+k$ sided dice whose k faces are labelled with *, and the remaining faces are numbered from 1 to m. Each face is equally likely to appear. The adversary now looks at all the outcomes and assigns weights from $[1, \cdots m]$ to the *'ed elements, trying to ensure that the minimum weight set in F is not unique. The problem is to place good bounds on m and k so that despite the adversary, the minimum weight set in F is unique with probability at least half.

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* [KVV]. 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?' [KVV].

Acknowledgements: We are thankful to David Aldous, Jack Edmonds, Laszlo Lovasz, Eva Tardos and Les Valiant for valuable discussions.

References

[AHU] A.V. Aho, J.E. Hopcroft, and J.D. Ullman, *The Design and Analysis of Computer Algorithms,* Addison-Wesley, Reading, Mass., 1974.

[BCP] A. Borodin, S.A. Cook, and N. Pippinger, 'Parallel Computation for Well-endowed Rings and Space Bounded Probabilistic Machines,' *Information* and *Control* 58, 1-3 (1983), pp. 113-136.

[BGH] A. Borodin, J. von zur Gathen, and J.E. Hopcroft, 'Fast Parallel Matrix and GCD Computations', *Twenty Third Annual IEEE Symp. on the Foundations of Computer Science (1982), pp. 65-71.*

[Br] A.Z. Broder, 'How Hard is it to Marry at Random? (On the Approximation of the Permanent)', *Eighteenth Annual Symp. on the Theory of Computing,* (1986), pp.

50-58.

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

[Cs] L. Csanky, 'Fast Parallel Matrix Inversion Algorithms,' *SIAM J. Computing* 5, (1976), pp. 618-623.

[Ed1] J. Edmonds, 'Paths, Trees and Flowers,' *Canad. J. Math.*, 17, (1965), pp. 449-467.

[Ed2] J. Edmonds, 'Systems of Distinct Representatives and Linear Algebra,' *J. Res. Nat. Bureau of Standards*, 71B, 4, (1967), pp 241-245.

[GP] 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), pp. 490-495.

[JVV] M.R. Jerrum, L.G. Valiant and V.V. Vazirani, 'Random Generation of Combinatorial Structures from a Uniform Distribution', *Theoretical Computer Science* 43 (1986) pp. 169-188.

[Ka] H. Karloff, 'A Randomized Parallel Algorithm for the Odd Set Cover Problem,' to appear in *Combinatorica*.

[KUW1] R.M. Karp, E. Upfal, and A. Wigderson, 'Constructing a Maximum Matching is in Random NC,' *Combinatorica*, 6(1), (1986) pp. 35-48.

[KUW2] R.M. Karp, E. Upfal, and A. Wigderson, 'Are Search and Decision Problems Computationally Equivalent?' *Seventeenth Annual Symp. on Theory of Computing*, (1985).

[KVV] 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), to appear in *Theoretical Computer Science*.

[Lo1] L. Lovasz, 'On Determinants, Matchings and Random Algorithms,' *Fundamentals of Computing Theory*, edited by L.Budach, Akademia-Verlag, Berlin, (1979).

[Lo2] L. Lovasz, '*Combinatorial Problems and Exercises*,' Akademiai Kaido, Budapest, and North-Holland, Amsterdam, (1979).

[LP] L. Lovasz and M. Plummer, *Matching Theory*, Academic Press, Budapest, Hungary, (in press).

[MV] 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), pp 17-27.

[Pa] 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).

[PY] C.H. Papadimitriou and M. Yannakakis, 'The Complexity of Restricted Spanning Tree Problems, *JACM*, Vol 29, No. 2, (1982), pp. 285-309.

[RV] M.O. Rabin and V.V. Vazirani, 'Maximum Matching in General Graphs Through Randomization,' submitted for publication.

[SV] M. Santha and U.V. Vazirani, 'Generating Quasi-Random Sequences from Semi-Random Sources', *JCSS*, Vol 33, No 1, Aug 1986, pp. 75-87.

[Sc] J.T. Schwartz, 'Fast Probabilistic Algorithms for Verification of Polynomial Identities,' *JACM*, 27(4), 701-717 (1980).

[Tu] W.T. Tutte, 'The Factorization of Linear Graphs,' *J. London Math. Soc.* 22, (1947), pp. 107-111.

[Va] L.G. Valiant, 'The Complexity of COmputing the Permanent', *Theoretical Computer Science* 8 (1979) pp. 189-201.

[ValVaz]L.G. Valiant and V.V. Vazirani, 'NP is as Easy as Detecting Unique Solutions,' *Theoretical Computer Science*, 47 (1986), pp. 85-93.

[Vaz] U.V. Vazirani, 'Randomness, Adversaries and Computation', Ph.D. Dissertation, University of California, Berkeley (1986).

[VV] U.V. Vazirani and V.V. Vazirani, 'The Two-Processor Scheduling Problem is in Random NC,' *Seventeenth Annual Symp. on Theory of Computing*, (1985), pp. 11-21.