CONSTRUCTING A PERFECT MATCHING IS IN RANDOM NC

Richard M. Karp†

Computer Science Division
University of California at Berkeley

Eli Upfal‡

Computer Science Department
Stanford University

Avi Wigderson††

IBM San Jose Research Laboratory

1. INTRODUCTION

In this paper we show that the problem of constructing a perfect matching in a graph is in the complexity class Random NC: i.e., the problem is solvable in polylog time by a randomized parallel algorithm using a polynomial-bounded number of processors. We also show that several related problems lie in Random NC. These include:

(i) Constructing a perfect matching of maximum weight in a graph whose edge weights are given in unary notation;

(ii) Constructing a maximum-cardinality matching;

(iii) Constructing a matching covering a set of vertices of maximum weight in a graph whose vertex weights are given in binary;

(iv) Constructing a maximum $s-t$ flow in a directed graph whose edge weights are given in unary.

Our results are based on a theorem of Tutte [Tu] showing that a graph has a perfect matching if and only if the determinant of a certain skew-symmetric matrix with indeterminates as elements is not identically zero. Around 1979 Lovasz suggested that Tutte's Theorem, combined with a fundamental randomized technique for testing whether a matrix with polynomial entries has a nonzero determinant [Sc], provides a simple polynomial-time randomized algorithm for testing whether a graph has a perfect matching. In 1982 Borodin, von zur Gathen and Hopcroft [BGH] observed that, since the problem of computing the determinant of a numerical matrix lies in NC, a parallel algorithm can be constructed based on Lovasz's approach. This algorithm establishes that the problem of *deciding* whether a graph contains a perfect matching lies in the complexity class Random NC, but leaves open the question of whether a

† Research supported by NSF Grant # DCR-8411954.
‡ Research supported by a Weizmann Post-Doctoral fellowship, and by DARPA Grant N00039-82-C-1036.
†† Research supported in part by DARPA Grant N00039-82-C-0235.

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.

© 1985 ACM 0-89791-151-2/85/005/0022 $00.75

parallel algorithm of comparable efficiency exists for *constructing* a perfect matching in a graph that is known to have such a matching. In 1984 Rabin and Vazirani [RV], using clever algebraic techniques related to Tutte's Theorem, gave an attractive randomized polynomial-time *sequential* algorithm for constructing a maximum matching. They also observed that *if* a graph has a *unique* perfect matching then the problem of finding it lies in NC. The special case of finding a perfect matching in an interval graph was shown to be in Random-NC by Kozen, Vazirani and Vazirani [KVV].

The additional ingredient that allowed us to obtain a Random-NC *parallel* algorithm for constructing a perfect matching in an arbitrary graph was the introduction of a useful set function called *Rank*. Our result is based on a Random-NC reduction of the matching problem to the problem of computing the *Rank* function, and on a Random-NC algorithm for computing the *Rank* of sets.

2. THE PERFECT MATCHING ALGORITHM

We present the perfect matching algorithm in three stages. First we reduce the problem of finding a perfect matching to the problem of identifying a large set of redundant edges in a graph that has a perfect matching and is not very sparse. We say that a set of edges in a graph that has a perfect matching is redundant if the removal of these edges results in a graph that still has a perfect matching. We then show that a random-NC procedure can construct a large set of redundant edges provided that a certain integer-valued function, called the rank, which is defined over all subsets of edges in the graph, is computable in random-NC. Finally we give a random-NC algorithm for computing the rank function.

2.1. A HIGH LEVEL DESCRIPTION OF THE ALGORITHM

Let $IG = (IV, IE)$ denote the input graph. The procedure Find_Perfect_Matching uses three main variables M, V and E.

The set variable M collects the edges that the procedure designates for the output matching. In contrast with the classical matching algorithms, our procedure never retreats from partial solutions. Once an edge is added to the set M it remains there until the procedure terminates, and it appears in the final output.

Throughout the execution of the procedure the variable V stores the set of vertices that are not yet covered by the matching M. The set E stores a subset of the set of edges connecting vertices in V; these are the edges that are still candidates for inclusion in the matching.

We start the procedure with $M = \emptyset$, $V = IV$, and $E = IE$. If the input graph IG has a perfect matching then with probability $1 - o(1)$ the following property hold throughout the execution of the procedure:

The graph $G = (V,E)$ has a perfect matching.

If this condition is ever violated the procedure will fail to produce a perfect matching in IG. On the other hand, if the condition remains true then it suffices for the algorithm to consider edges in the set E, since the set M together with a perfect matching in G is clearly a perfect matching in the input graph IG.

```
procedure Find_Perfect_Matching(V,E,M);
while |E| ≥ 0 do
  if |E| < (3/4)|V| then [G is sparse]
    find a set NM, |NM| ≥ (1/4)|V|, of edges that lie in
      every perfect matching;
    M ← M ∪ NM;
    V ← V − { vertices covered by NM };
    E ← {(v,u)|(v,u)∈E, v,u∈V};
  else [G is not sparse]
    Find_Redundant_Set(RE);
    {with probability β > 0, |RE| ≥ α|E|}
    E ← E − RE;
```

end.

The main while loop of the procedure is executed until the set E is empty. Assuming that the graph (V,E) always has perfect matching, this implies that the set V is also empty, thus, that the variable M stores a perfect matching in G.

Each iteration of the while loop tries to decrease the size of the set E by a constant factor. The execution distinguishes between two cases:

Case a: $|E| < \frac{3}{4}|V|$, (the graph $G = (V,E)$ is sparse). Since G has a perfect matching, the degrees of all its vertices are at least 1. However, $|E| < \frac{3}{4}|V|$, thus at least $\frac{1}{2}|V|$ vertices have degree exactly 1. If an edge is incident to a vertex with degree 1 this edge is included in any perfect matching in this graph. Thus we can identify at least $\frac{1}{4}|V|$ edges that can be added to the set M and the remaining graph (V',E'), defined by the vertices in V that are not covered by the new matching edges, has a perfect matching. Furthermore, $|E'| \leq |E| - \frac{1}{4}|V| \leq \frac{7}{8}|E|$, and the computation can be done in $O(1)$ steps using $O(|E|)$ processors.

Case b: $|E| \geq \frac{3}{4}|V|$. This case, which is the novel part of the algorithm, is solved by the probabilistic procedure **Find_Redundant_Set** . With probability $1 - o(\frac{1}{|E|^2})$ the procedure produces a set RE such that the graph $(V,E-RE)$ has a perfect matching. With probability $\beta > 0$ $|RE| \geq \alpha|E|$ for some $\alpha > 0$. Thus with probability $\beta > 0$ the procedure reduces the size of our problem by a constant factor.

The analysis of the number of iterations of the while loop required to create a perfect matching requires a fact from probability theory.

FACT 1 [AV] *Let X be a random variable distributed as the number of successes in n independent Bernoulli trials with probability of success p. If $0 < a < 1$ then $P[X \leq (1-a)np] < \exp(-\frac{1}{2}a^2np)$.*

LEMMA 2.1: *If there are constants $\alpha,\beta > 0$ such that with probability at least β a call to the procedure* Find_Redundant_Set *identifies at least a fraction α of the edges in E as redundant, then, uniformly for all problem instances, the expected number of iterations of the while loop within this procedure is $O(\log E)$, and, moreover, for some $c > 0$ and $d > 0$ the number of iterations of the while loop is bounded above by $d \log E$ with probability $1 - O(E^{-c})$.*

Proof: We may assume without lost of generality that $\alpha \leq \frac{1}{3}$. Call an iteration of the while loop a success if it disposes of at least αE edges, either by determining that they lie in every perfect matching (Case a) or determining that they are redundant (Case b). Then every iteration in which Case a occurs is necessarily a success, and every iteration in which Case b occurs is a success with probability at least β. Let $\eta = (1-\alpha)^{-1}$. The number of successes required to reduce the number of edges to 1, starting from an initial edge set E, is at most $\log_\eta |E|$, and hence the number of successes required to complete the entire computation is at most $1 + \log_\eta |E|$. Since each iteration has probability of success at least β, independently of the outcomes of all previous iterations, the expected number of iterations is at most $\beta^{-1}(1 + \log_\eta |E|)$. Moreover, the number of successes in the first $2\beta^{-1}\log_\eta |E|$ iterations is a random variable that stochastically dominates the number of successes in $2\beta^{-1}\log_\eta |E|$ independent Bernoulli trials with probability of success β. Applying Fact 1, we find that the probability of having $\log_\eta |E|$ or fewer successes in the first $2\beta^{-1}\log_\eta |E|$ iterations is at most $\exp(-\frac{1}{4}\log_\eta |E|) = |E|^{-c}$, where $c = \frac{1}{4\ln \eta} > 0$. Hence, with probability $1 - |E|^{-c}$ procedure Find_Perfect_Matching terminates within $2\beta^{-1}\log_\eta |E|$ iterations.

Proof: Let $|V| = n$, $|E| = m$, and let p_l denote the probability that for a set S drawn at random from the l-element subsets of E, and for an edge drawn at random from $E - S$, $Rank(S \cup \{e\}) = Rank(S) + 1$.

We can think of a random set S as being constructed by a sequence of l random choices (without repetitions) from the set E. $Rank(S)$ is then equal to the number of times the *Rank* was increased when a new random edge was added to the set S. Thus,

$$\mathop{E}_{|S|=l}[Rank(S)] = \sum_{j=0}^{l-1} p_j.$$

Obviously $Rank(E) = \frac{n}{2}$, since the graph has a perfect matching, so we can write

$$\sum_{j=1}^{\frac{1}{6}m} p_j \le \sum_{j=0}^{m-1} p_j = Rank(E) = \frac{n}{2} \le \frac{4}{6}m$$

Using this upper bound on $\sum p_j$ we can compute a lower bound for the average size of the set RE produced by our procedure. The size l of the set S is chosen uniformly from the range $1, \ldots, \frac{4}{6}m$. When a random set S of size l is chosen, the probability that a random edge $e \in E - S$ is added to the set RE is $1 - p_l$. There are always at least $\frac{1}{6}m$ edges in $E - S$, therefore

$$\mathbb{E}[|RE|] \ge \frac{1}{\frac{4}{6}m} \sum_{l=1}^{\frac{4}{6}m} (1 - p_l)\frac{1}{6}m \ge$$

$$\frac{1}{5}\left(\frac{5}{6}m - \sum_{l=1}^{\frac{4}{6}m} p_l\right) \ge \frac{1}{5}\frac{1}{6}m.$$

Let $\mu = \frac{1}{2}\frac{1}{5}\frac{1}{6}$, then

$$2\mu m = \mathbb{E}[|RE|] = \sum_{j=1}^{m} j(Prob\{|RE| = j\}) \le$$

$$\mu m(Prob\{|RE| < \mu m\}) + m(Prob\{|RE| \ge \mu m\}),$$

which implies that

$$2\mu m \le \mu m + mProb\{|RE| \ge \mu m\},$$

or

$$Prob\{|RE| \ge \frac{1}{60}m\} \ge \frac{1}{60}.$$

LEMMA 2.4: *If $Rank(S)$ is computable in $f_1(|E|)$ steps using $h_1(|E|)$ processors, where f_1 and h_1 are monotone nondecreasing functions, then* Find_Redundant_Set *is computable in $f_1(|E|) + O(1)$ steps using $|E|h_1(|E|)$ processors.*

Proof: The simultaneous computation of $Rank(S)$ and of $Rank(S \cup \{e\})$ for all e in $E - S$ requires $f_1(|E|)$ steps using $|E|h_1(|E|)$ processors. The determination of RE once these *Rank* computations have been done requires $O(1)$ steps using $O(|E|)$ processors.

2.3. COMPUTING THE RANK FUNCTION

For simplicity of presentation we first give a solution for the case where the input graph is bipartite. We then explain how the general case is handled.

Let $G = (U, V, E)$ denote a bipartite graph with edge set E and n vertices in each part. Let $U = \{u_1, u_2, \ldots, u_n\}$ be the vertex set of one part and $V = \{v_1, v_2, \ldots, v_n\}$, the vertex set of the other part. Associate with each edge $\{u_i, v_j\}$ an indeterminate x_{ij}, and let $B = (b_{ij})$ be a $n \times n$ matrix of indeterminates defined by the following rule:

$$b_{ij} = \begin{cases} x_{ij} & \text{if } \{u_i, v_j\} \in E \\ 0 & \text{if } \{u_i, v_j\} \notin E. \end{cases}$$

Edmonds [Ed] has observed that G has a perfect matching if and only if $det(B) \not\equiv 0$. This is true because each of the $n!$ terms in the sum $\sum_{\sigma \in S_n} sign(\sigma) \prod b_{i,\sigma(i)}$ is a product of n entries in B that correspond to the edges of one of the $n!$ perfect matchings in the complete bipartite graph. If all the edges of a perfect matching

exist in G then the corresponding term in $det(B)$ is not identically zero; otherwise one of the entries is zero and $\Pi b_{i,\sigma(i)} = 0$. It is easy to verify that monomials can not cancel each other in this summation, and therefore $det(B) \not\equiv 0$ if and only if G has a perfect matching.

To compute the *Rank* of a given set $S \subseteq E$ we refine this argument. Define the matrix $B[S] = \left[b_{ij}\right]$ as follows:

$$b_{ij} = \begin{cases} yx_{ij} & \text{if } \{u_i, v_j\} \in S \\ x_{ij} & \text{if } \{u_i, v_j\} \in E - S \\ 0 & \text{if } \{u_i, v_j\} \notin E. \end{cases}$$

Le $B[S]$ is derived from the matrix B by tagging all entries that correspond to edges in S with a new variable y.

Each non-zero monomial in $det(B[S])$ corresponds to a perfect matching in G. The degree in y of a monomial is equal to the number of elements from S that participate in the corresponding perfect matching. Again monomials can not cancel each other, hence the degree in y of $det(B[S])$ is equal to that of the monomial with the maximum degree in y, which in turn is equal to the maximum number of elements from S that participate in a perfect matching in G, i.e., to $Rank(S)$.

So in order to compute $Rank(S)$ we have to compute the degree in y of the multivariate polynomial $det(B[S])$. Rewriting $det(B[S])$ as a polynomial in y, we get $det(S) = \sum_{i=0}^{n} Q_i[\{x_{ij}\}]y^i$, and $Rank(S) = Max\{i \mid Q_i \not\equiv 0\}$. Unfortunately, we can not test directly whether Q_i, which is a polynomial in up to $|E|$ indeterminates, is identically zero. Instead, we use a well-known probabilistic method.

THEOREM 2.1: (Schwartz) [Sc] *Let $\tilde{Q}_i$ denote the value of the polynomial Q_i when each indeterminate in Q_i is replaced by a random integer in the range $1, \ldots, J$. If $Q_i \not\equiv 0$ then $Prob\{\tilde{Q}_i = 0\} \leq \frac{|E|}{J}$.*

Let $\tilde{B}[S]$ denote the matrix $B[S]$ after all its indeterminates, except y, have been replaced by random integers. To compute the $\tilde{Q}_i$'s we have to compute $det(\tilde{B}[S])$ as a polynomial in one variable y.

THEOREM 2.2: (Borodin, Cook, Pippenger) [BCP]. *The determinant of an $n \times n$ matrix of polynomials with a constant number of variables and the degree of each matrix element bounded by n can be computed in $O(\log^2 n)$ steps using $O(n^{4.5})$ processors.*

We can now summarize the algorithm for computing the *Rank* function in the bipartite case.

```
procedure Rank(S);
begin
  construct the matrix B[S] = [b_ij];
  replace each indeterminate x_ij by a random integer in
    the range 1, ..., |E|^4;
  compute det(B̃[S]) = Σ_i Q̃_i y^i;
  Rank(S) = Max{i | Q̃_i ≠ 0};
end.
```

In the case of a general graph our method of computing the rank function relies on the following theorem of Tutte.

THEOREM 2.3 (Tutte) [Tu] *Let $G = (V,E)$ denote a general graph with vertex set $V = \{1,2,\ldots,n\}$ and define the skew-symmetric matrix $B = \left[b_{ij}\right]$ as follows:*

$$b_{ij} = \begin{cases} x_{ij} & \text{if } \{i,j\} \in E \text{ and } i<j \\ -x_{ij} & \text{if } \{i,j\} \in E \text{ and } i>j \\ 0 & \text{if } \{i,j\} \notin E. \end{cases}$$

Then G has a perfect matching if and only $det(B) \not\equiv 0$.

Tutte's theorem can be extended to yield an algorithm for computing the *Rank* function in general graphs.

Sep.12. 2002 12:28PM No. 7761 P. 8/13

THEOREM 2.4: *Let $G = (V,E)$, $S \subseteq E$, and define the matrix $B[S] = \left[b_{ij}\right]$ as follows:*

$$b_{ij} = \begin{cases} yx_{ij} & \text{if } \{i,j\} \in S \text{ and } i<j \\ -yx_{ij} & \text{if } \{i,j\} \in S \text{ and } i>j \\ x_{ij} & \text{if } \{i,j\} \in E-S \text{ and } i<j \\ -x_{ij} & \text{if } \{i,j\} \in E-S \text{ and } i>j \\ 0 & \text{if } \{i,j\} \notin E. \end{cases}$$

Then Rank (S) is equal to half of the degree in y of $\det(B[S])$.

Proof: Let P be the set of all permutations σ of $\{1,2,\ldots,n\}$ such that, for all i, $b_{i,\sigma(i)} \neq 0$. We classify these permutations according to their cycle structures. Since all elements on the main diagonal are 0, no permutation in P contains a cycle of length 1. Let OP be the set of all permutations in P which contain at least one odd cycle and let EP be the set of all permutations in S in which all cycles are of even length. Then P is the disjoint union of OP and EP. Let M be the set of all permutations in P such that every cycle is of length 2. Then $M \subseteq EP$, and the permutations in M are in one-to-one correspondence with the perfect matchings in G.

We have $\det(B[S]) = \sum_{\sigma \in P} \operatorname{sign}(\sigma) \prod_i b_{i,\sigma(i)}$. We shall show that the contribution of the permutations containing odd cycles to $\det(B[S])$ is equal to zero; i.e., that $\sum_{\sigma \in OP} \operatorname{sign}(\sigma) \prod_i b_{i,\sigma(i)} = 0$. We shall partition the permutations containing odd cycles into pairs, such that the two permutations in each pair make a net contribution of zero to the determinant. Let σ be a permutation containing at least one odd cycle. Let l be the least element of $\{1,2,\ldots,n\}$ occurring in an odd cycle of σ, and let C be the odd cycle containing l. Then σ is paired with a permutation $\bar{\sigma}$ which is the same as σ except that the cycle C is reversed. Thus $\bar{\sigma}$ is defined as follows: if $j \notin C$ then $\bar{\sigma}(j) = \sigma(j)$; if $j \in C$ then $\bar{\sigma}(j) = \sigma^{-1}(j)$. It is easy to verify that this rule partitions OP into pairs, and that each pair makes a net contribution of zero to the determinant.

Call an ordered pair $\langle i,j \rangle$ an $S-$pair if $\{i,j\}$ is an edge in S. Then the $S-$pairs correspond to the entries in the matrix $B[S]$ which involve the variable y. Associated with each permutation σ in EP is the term $\operatorname{sign}(\sigma) \prod_i b_{i,\sigma(i)}$. This term is a nonzero monomial in the variables $\{x_{ij}\}$ and y. Define the $y-$degree of σ as the degree of y in this monomial. Then the $y-$degree of σ is just the number of $S-$pairs in σ, i.e. the number of $S-$pairs $\langle i,\sigma(i) \rangle$.

We shall show that, for every permutation $\sigma \in EP$ there exists a permutation $\tau \in M$ such that the $y-$degree of τ is greater than or equal to the $y-$degree of σ. To construct τ, partition the pairs $\langle i, \sigma(i) \rangle$, $i = 1,2,\ldots,n$ into two sets, called the odd pairs and the even pairs, in such a way that $\langle i, \sigma(i) \rangle$ is an odd pair if and only if $\langle \sigma(i), \sigma(\sigma(i)) \rangle$ is an even pair. In other words, the partition is chosen so that, in the traversal of any cycle of σ, odd pairs and even pairs alternate. Such a partition is possible because all the cycles in σ are even. Assume without loss of generality that the set of odd pairs in σ contains at least as many $S-$pairs as does the set of even pairs in σ. If the $y-$degree of σ is d, then the set of odd pairs contains at least $\frac{d}{2}$ $S-$pairs. Now define τ by the following rule: if $\langle i, \sigma(i) \rangle$ is an odd pair then $\tau(i) = \sigma(i)$ and $\tau(\sigma(i)) = i$. Then τ lies in M and the $y-$degree of τ is at least d.

Each permutation in M corresponds to a perfect matching in G, and the $y-$degree of this permutation is twice the number of edges from S in this perfect matching. Hence, using the result proven in the last paragraph, the maximum $y-$degree of any permutation is just twice the rank of S. To show that the degree of y in $\det(B[S])$ is twice the rank of S, we need to show that the permutations of maximum $y-$degree make a nonzero net contribution to $\det(B[S])$. But this is clear, since there exists a permutation in M among those of

maximum y-degree, and its monomial is not cancelled by the monomial of any other permutation.

Thus, using the probabilistic method of Schwartz and the algorithm of Borodin, Cook and Pippenger, we have a Random-NC algorithm for computing the rank of a set of vertices in a general graph.

LEMMA 2.5:

1. *The procedure* Rank *is executed in* $O(\log^2|V|)$ *steps using* $O(|V|^{4.5})$ *processors.*
2. *The probability that the procedure* Rank *fails to compute the correct value of* $Rank(S)$ *is bounded by* $O(|E|^{-2})$, *and this event does not depend on the input.*

Combining now the results of lemmas 2.1-2.5 we have

THEOREM 2.6:

For any input graph $G = (V,E)$:

1. *The procedure* Find_Perfect_Matching *uses* $O(|V|^{4.5})$ *processors and terminates within* $O(\log^2|E|)$ *steps with probability* $1 - \frac{1}{|E|^c}$ *for some* $c>0$.
2. *The probability that the procedure fails to produce a perfect matching when applied to a graph that possesses one is bounded by* $\frac{1}{|E|^c}$.

3. FURTHER RESULTS

In this section we derive Random-NC algorithms for several further problems related to matching and network flows. We begin by giving such an algorithm for finding a perfect matching of maximum weight in an edge-weighted graph $G = (V,E,\omega)$, when the edge-weights $\omega(\{i,j\})$ are given in unary.

Definition 3.1: *Let* MW *denote the set of perfect matchings of maximum weight in the weighted graph* $G = (V,E,\omega)$. *For any set* $S \subseteq E$ *define*

$$Rank_w(S) = \max_{A \in MW} |S \cap A|.$$

In words $Rank_w(S)$ is an integer giving the maximum number of edges from S that participate in a perfect matching of maximum weight in G.

It is easy to verify that running the procedure Find_Perfect_Matching with the new rank function computes the desired perfect matching in an expected number of iterations of order $\log|E|$. The only difficulty is to show that the new $Rank_w$ function is computable in Random NC. The following theorem, in combination with Theorem 2.2, establishes this fact.

THEOREM 3.1: *Let* $G = (V,E,\omega)$, $S \subseteq E$, *and define the matrix* $B[S]$ *as follows:*

$$b_{ij} = \begin{cases} yz^{w}x_{ij} & \text{if } \{i,j\} \in S,\ i<j, \text{ and } \omega(\{i,j\})=w \\ -yz^{w}x_{ij} & \text{if } \{i,j\} \in S,\ i>j, \text{ and } \omega(\{i,j\})=w \\ z^{w}x_{ij} & \text{if } \{i,j\} \in E-S,\ i<j, \text{ and } \omega(\{i,j\})=w \\ -z^{w}x_{ij} & \text{if } \{i,j\} \in E-S,\ i>j, \text{ and } \omega(\{i,j\})=w \\ 0 & \text{if } \{i,j\} \notin E \end{cases}$$

Let $\det(B[S]) = \sum_i Q_i z^i$, *and let* $L = \max_i\{i \mid Q_i \not\equiv 0\}$.

1. *The the maximum weight of a perfect matching in* G *is* $\frac{L}{2}$.
2. $Rank_w(S)$ *is equal to half the degree in* y *of* Q_L, *the coefficient of* z^L.

The proof of Theorem 3.1 is quite similar to that of Theorem 2.4. First it is shown that the permutations containing odd cycles make a net contribution of zero to $\det(B[S])$. Then attention is restricted to EP, the set of permutations with all cycles even that make a nonzero contribution to $\det(B[S])$. Let L be the highest degree to which z occurs in the monomial associated with such a permutation, and let d be the highest degree of y that occurs in a monomial that is of degree L in z. It is shown that, among the permutations whose monomials are of degree L in z and of degree d in y, there is at least one whose cycles are all of length 2. Such a permutation is shown to correspond to a matching of weight

$\frac{L}{2}$ containing $\frac{d}{2}$ edges from S; moreover, it is shown that this matching is of maximum weight and, among matchings of maximum weight, has a maximum number of edges from S. Finally, it is shown that the monomials of degree L in x and d in y associated with permutations in EP make a nonzero net contribution to $\det(B[S])$, so that the polynomial $\det(B[S])$ is of degree L in x, and the coefficient of x in this polynomial is of degree d in y.

In the following paragraphs we use the technique of reducibility to show that further matching and flow problems lie in Random NC. All the reductions mentioned below can be performed in logspace.

1. Maximum cardinality matching: The problem of constructing a maximum cardinality matching in a graph G with vertex set V and edge set E is easily reduced to the problem of constructing a perfect matching of maximum weight in a graph G' with vertex set V in which each edge has weight zero or one. We can assume that $|V|$ is even. In this reduction G' is the complete graph on vertex set V; edge $\{i,j\}$ of the complete graph receives weight one if $\{i,j\}$ lies in E, and weight zero otherwise. This reduction shows that the maximum cardinality matching problem lies in Random-NC (A different reduction for this problem is given in [KV]).

2. Vertex weighted matching: The vertex-weighted matching problem is in Random NC even when the weights of the vertices are given in binary notation. In this problem we are given a graph G with vertex set V in which each vertex v has a positive weight $w(v)$. We seek a matching that covers a set of vertices of maximum total weight.

This problem can by approached with the help of matroid theory. The results from matroid theory that we require can be found in the comprehensive reference [We] Call a set of vertices S *independent* if there is a matching that covers all the vertices in S; then our goal is to construct an independent set of maximum weight. Let I be the family of all independent sets. The structure (V,I) is a matroid. In this matroid, the rank of a set of vertices S is just the maximum number of vertices from S that are covered by a matching. A maximum-weight independent set T in a matroid can be constructed by the following rule: let the elements be $v_1,v_2,...,v_{|V|}$ in order of decreasing weight; then, for $i = 1,2,...,|V|$, v_i lies in T if and only if $Rank(\{v_1,v_2,...,v_i\}) > Rank(\{v_1,v_2,...,v_{i-1}\})$. Once T is known, the desired matching is constructed by finding a perfect matching in the subgraph of G induced by T, using the main algorithm of this paper.

Thus, a Random-NC algorithm for the vertex-weighted matching problem is at hand; we also give a Random-NC algorithm for computing the rank of a set of vertices in this matroid. But the problem of computing $Rank(S)$ is easily reduced to that of determining a maximum-weight perfect matching in a graph K with vertex set V whose edges are of weight 0,1 and 2. We can assume without loss of generality that the given graph $G = (V,E)$ has an even number of vertices. The graph K is the complete graph on vertex set V. If edge $\{i,j\}$ does not lie in E then $\{i,j\}$ is given weight 0; if $\{i,j\}$ lies in E, then the weight of $\{i,j\}$ is $|S \cap \{i,j\}|$. Clearly, the maximum weight of a perfect matching in K is the rank of S. Thus we have shown that the vertex-weighted matching problem lies in Random NC.

3. Network flow: First, consider the problem of constructing a maximum $s-t$ flow in a directed graph in which each edge has capacity 1. There is a classical reduction of this problem to the problem of constructing a maximum matching in a bipartite graph, as follows. Let the flow network be $G = (V,E)$, with source s and sink t. We may assume that s has in-degree 0 and t has out-degree 0. The reduction constructs a bipartite graph H with bipartition (V_1, V_2). Each part of the bipartition is a copy of the edge set of G. Thus,

$V_1 = \{(v,1) \mid e \in E\}$ and $V_2 = \{(e,2) \mid e \in E\}$. If the head of edge e is also the tail of edge f (i.e., $e = (i,j)$ and $f = (j,k)$ for some i,j and k) then H contains and edge between $(e,1)$ and $(f,2)$. If an edge e in G is incident with neither s nor t, then H has an edge from $(e,1)$ to $(e,2)$. Then a maximum matching in H yields a maximum flow in G according to the following rule: edge e carries a flow of 1 if and only if $(e,1)$ is matched with some vertex $(f,2)$, where $e \neq f$, or $(e,2)$ is matched with some vertex $(f,1)$, where $e \neq f$. The reduction just given extends easily to the case in which the flow network has edges with integer capacities, provided these capacities are given in unary. The idea is to replace each edge (i,j), of capacity c, with c parallel edges from i to j, each of capacity 1. All capacities in the resulting network are 1, and thus the reduction to bipartite matching applies. Thus, we have shown that the following problem is in Random NC: construct a maximum $s-t$ flow in a directed flow network whose edge capacities are given in unary.

We summarize the constructions and reductions given in this section by a theorem.

THEOREM 3.2: *The following problems lie in Random-NC:*

(i) *Constructing a perfect matching of maximum weight in a graph whose edge weights are given in unary.*

(ii) *Constructing a maximum matching;*

(iii) *Constructing a matching covering a set of vertices of maximum weight in a graph whose vertex weights are given in binary;*

(iv) *Constructing a maximum $s-t$ flow in a directed graph whose edge weights are given in unary.*

Our result about network flows stands in interesting contrast to the following result due to Goldschlager, Shaw and Staples [GSS]: the problem of constructing a maximum $s-t$ flow in a directed flow network with edge capacities given in binary is complete in P with respect to logspace reductions. Since it is generally believed that such complete problems do not lie in Random-NC, it appears that the parallel complexity of the max-flow problem depends critically on whether the capacities are given in unary or in binary. Nevertheless, the following result can be given:

THEOREM 3.3: *There is a randomized parallel algorithm to construct a maximum $s-t$ flow in a directed network whose edge weights are given in binary, such that the number of processors used is bounded by a polynomial in the number of vertices, and the time used is $O((\log V)^k \log C)$, where C is the largest capacity of any edge and k is a constant.*

This result is proved by combining the techniques of the present paper with the Edmonds-Karp scaling technique [EK]. Of course the result does not place the problem in Random NC, since $\log C$ is a linear, rather than polylogarithmic, function of the number of bits needed to express C in binary.

4. DISCUSSION

Each of the randomized algorithms given in this paper has a small probability of giving an erroneous result: for example, procedure Find_Perfect_Matching may fail to produce a perfect matching in a graph that possesses one, and therefore its failure does not indicate with certainty that no perfect matching exists. Running the algorithm many times in parallel can reduce the probability of error to an exponentially low level, but can never eradicate it entirely. Howard Karloff [Kar] has given a Random NC algorithm for the odd-set cover problem, which is the "dual" of the matching problem. As Karloff points out, this result can be combined with our algorithms to yield algorithms for the perfect matching problem and the maximum matching problem which run in polylog expected time and always give the correct result; i.e., Las Vegas algorithms rather than Monte Carlo algorithms.

Sep.12. 2002 12:30PM NO.776 P. 12/13

It remains an open question whether randomization can be dispensed with entirely in these problems. It would be very nice to show that the problem of deciding whether a graph has a perfect matching lies in NC, and even nicer to show that the problem of constructing a perfect matching lies in NC.

Finally, the investigations reported here have led us into a broader study of the relation between decision problems and search problems; the results of that study are reported in the companion paper [KUW].

REFERENCES

[AV] D. Angluin and L.G. Valiant, Fast probabilistic algorithms for Hamiltonian circuits and matchings. *J. of Comp. Syst. Sci. 18*, (1979) pp. 155-193.

[BCP] A. Borodin, S.A. Cook, and N. Pippenger. Parallel computation for well-endowed rings and space bounded probabilistic machines. *Information and Control 58* 1-3 (1983) pp. 113-136.

[BOH] A. Borodin, J. von zur Gathen, and J. Hopcroft, Fast parallel matrix and GCD computations. *Proc. 23rd STOC* (1982) pp. 65-71.

[Co] S.A. Cook. An overview of computation complexity. *CACM* 26 (1983) pp. 400-408.

[Ed] J. Edmonds, Systems of distinct representatives and linear algebra. *J. of Res. Nat. Bureau of Standards*. 71A (1967) pp. 241-245.

[EK] J. Edmonds and R.M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. of ACM 19* (1972) pp. 248-264.

[GSS] L.M. Goldschlager, R.A. Shaw and J. Staples. The maximum flow problem is logspace complete for P. *Theoretical Computer Science 21* (1982) pp. 105-111.

[Kar] H.J. Karloff, A randomized parallel algorithm for the odd-set cover problem. Submitted, 1985.

[KUW] R.M. Karp, E. Upfal and A. Wigderson, Are search and decision problems computationally equivalent? *STOC 1985*.

[KVV] D. Kozen, U.V. Vazirani and V.V. Vazirani, The two-processor scheduling problem is in R-NC. *STOC 1985*.

[RV] M.O. Rabin and V.V. Vazirani, Maximum matchings in general graphs through randomization. TR-15-84, *Harvard University Center for Research in Computing Technology*, 1984.

[Sc] J.T. Schwartz, Fast probabilistic algorithms for verification of polynomial identities. *J. of ACM, 27* 4 (1980) pp. 701-717.

[SU] E. Shamir and E. Upfal, *N*-processor graphs distributively achieve perfect matching in $O(\log^2 N)$ beats. *Proceedings of the First ACM SIGACT-SIGMOD Symp. on Principles of Distributed Computing*. Ottawa, 1982, pp. 238-241.

[Tu] W.T. Tutte, The factors of graphs. *Canad. J. Math.* 4 (1952) pp. 314-328.

[We] D.J.A. Welsh, *Matroid Theory*. Academic Press (1976).

Sep.12. 2002 12:31PM NO.771 P. 13/13