THE MATCHING PROBLEM FOR BIPARTITE GRAPHS WITH POLYNOMIALLY BOUNDED PERMANENTS IS IN NC (EXTENDED ABSTRACT)

Dima Yu. Grigoriev
Steklov Institute of Mathematics
Sov. Acad. of Sciences
Leningrad 191011

Marek Karpinski
Dept. of Computer Science
University of Bonn
5300 Bonn 1

Abstract.

It is shown that the problem of deciding and constructing a perfect matching in bipartite graphs G with the polynomial permanents of their $n \times n$ adjacency matrices A ($perm(A) = n^{O(1)}$) are in the deterministic classes NC^2 and NC^3, respectively. We further design an NC^3 algorithm for the problem of constructing all perfect matchings (*enumeration problem*) in a graph G with a permanent bounded by $O(n^k)$. The basic step was the development of a new symmetric functions method for the decision algorithm and the new parallel technique for the matching enumerator problem. The enumerator algorithm works in $O(\log^3 n)$ parallel time and $O(n^{3k+5.5} \cdot \log n)$ processors. In the case of arbitrary bipartite graphs it yields an 'optimal' (up to the log n-factor) parallel time algorithm for enumerating all the perfect matchings in a graph. It entails also among other things an efficient NC^3-algorithm for computing small (polynomially bounded) arithmetic

permanents, and a sublinear parallel time algorithm for enumerating all the perfect matchings in graphs with permanents up to $2^{n^{\varepsilon}}$.

1. Introduction.

Given a bipartite graph G, and its (bipartite) adjacency matrix A. The problem of constructing all perfect matchings of G (the computation of the arithmetic permanent $perm(A)$) is $\#P$-complete [Va 79]. Let PER^T (*logical permanent problem*) denote the set of all square adjacency matrices that have a perfect matching. PER^T does have polynomial time algorithms and $O(m^5)$-uniform circuits [HK 73],[Ra 85].

Also a problem of finding some perfect matching (not the enumeration of all matchings) can be done in polynomial time [HK 73]. The problem of perfect matching for bipartite graphs is known to be in RNC^2 [MVV 87], [KUW 85]. The problem of deciding whether there exists a perfect matching (the problem of the logical permanent) possesses some interesting lower bound properties for monotone circuits [Ra 85], as well as interesting connections of its circuit upper bounds to the intractability of the discrete logarithm problem [FLS 85], for example. In 1984, Rabin and Vazirani [RV 84] have proved that if a graph has a unique perfect matching, then the problem of finding it lies in NC.

Kozen, Vazirani and Vazirani [KVV 85] and Hembold and Mayr [HM 86] have designed NC-algorithms for the problem of testing for unique matching as well as for interval graphs and the connected problem of 2-processor scheduling. [DK 86a] has generalized the result on interval graphs to strongly chordal graphs ([Fa 83], [Ta 85]). It was observed in [DK 86a] that the perfect matching for chordal graphs is complete for the general matching problem. Surprisingly, it was proved in [DK 86b] that the problem of matching for regular graphs is complete for the general matching problem.

It is also known that the perfect matching construction for bipartite regular graphs is in NC^2[LPV 81]. In [Br 86] interesting approximation methods have been proposed for bipartite matching problems. The status of the general perfect matching problem remains open and is still one of the most intriguing open problems in parallel computation.

In this paper we attack the problem of checking and constructing perfect matchings in bipartite graphs in the case where its number is bounded by the constants and the polynomials. It was known from Rabin and Vazirani [RV 84] that if a graph has a unique perfect matching, then the problem of finding it lies in NC.

The aim of this paper is to prove the following three results:

(1) If a bipartite graph G has a polynomial adjacency permanent ($perm(A_G) \leq$

cn^k), then the problem of deciding the existence of a perfect matching and its construction is in NC^2 and NC^3, respectively (Theorems 1 and 3).

(2) If a bipartite graph G has a bounded adjacency permanent ($perm(A_G) \leq k$), then the construction problem of 'all perfect matchings' lies in NC^2 (Theorem 2).

(3) If a bipartite graph G has a polynomial adjacency permanent, then the problem of constructing all perfect matchings lies in NC^3 (Theorem 4). The enumerator algorithm works within $O(\log^3 n)$ parallel time and $O(n^{3k+5.5}\log n)$ processors.

The algorithm involves development of the new method of symmetic functions (Theorem 1) and the new parallel techniques for the matching construction and the matching enumerator.

It is interesting to notice that we have displayed a new parallel complexity feature of the matching problem, the easiness of its parallel enumerator for the small number of solutions. This feature is seemingly not shared, on the different complexity levels, by other hard counting problems (cf. [VV 85], [MVV 87]).

2. The Algorithms.

Given a bipartite graph of n vertices, denote its 0-1 bipartite adjacency matrix by A_G. The *permanent* of G is the permanent of $A_G = (a_{ij})_{n\times n}$, i.e. the *number* $perm(A_G) = \sum_\sigma a_{1\sigma(1)}\ a_{2\sigma(2)}\ \cdots\ a_{n\sigma(n)}$, where summation extends over all permutations σ on $\{1, 2, \cdots, n\}$. Given a 0-1 matrix $A = (a_{ij})$, a *1-pattern* t_A of A is a mapping from $\{1, \cdots, n\} \times \{1, \cdots, n\}$ into $\{1, \cdots, n^2\}$ such that $t_A(i,j) = l$ if $a_{ij} = 1$ and a_{ij} is the l-th non-zero element, $t_A(i,j) = 0$ otherwise.

Theorem 1. If a bipartite graph G has a polynomial permanent, $perm(A_G) \leq cn^k$ for given constants c and k, then the problem of deciding the existence of a perfect matching (the *logical permanent problem)* is in NC^2.

PROOF. Suppose G is a given graph of n vertices and A its adjacency matrix. Let $p_1, p_2, \cdots, p_k$ denote *consecutive* prime numbers. We construct the following NC^2-algorithm for deciding the existence of a perfect matching:

1. Construct in parallel all matrices

$$A_m = (a_{ij}^m) \text{ for } 1 \leq m \leq cn^k \quad \text{by}$$

$$a_{ij}^m = \begin{cases} (p_{t_A(i,j)})^m & \text{if } a_{ij} = 1 \\ 0 & \text{otherwise} \end{cases}$$

2. Compute the determinants of A_m, $1 \le m \le cn^k$: $\quad Det(A_m) = \alpha_m$

In this paper we shall use the boolean circuit model of computation ([Co 85]). Computing the determinants of an $n \times n$ matrix of n-bits numbers takes $O(\log^2 n)$ boolean parallel time and $O(n^{4.5})$ processors ([BGH 82], [BCP 83]).

3. **If** $\exists m[\alpha_m \neq 0]$ **then** *'accept'* **else** *'reject'*.

We prove the correctness of the algorithm by the following

Lemma 1. $\forall m[\alpha_m = 0] \Longleftrightarrow perm(A) = 0$.

PROOF. ($\Longrightarrow$). We make use of the fact that the determinants of the consecutive matrices (a_{ij}^m) form symmetric differences of the form $\sum_i x_i^m - \sum_j y_j^m$, for x_i, y_j prime codings of all matchings, $1 \le m \le cn^k$. Codings x_i and y_j are pairwise different $x_i \neq y_j$, $x_i \neq x_j$, etc. or equal to zero. $\sigma_m = \sum_i x_i^m$ and $\sigma'_m = \sum_j y_j^m$ are symmetric functions.
All such functions are uniquely represented by the elementary symmetric functions s_i (cf. [Ma 79]), s_i stands for the i-th symmetric function, by the use of the Newton formulas (cf. [Ga 60], pp. 87-88): $\sigma_1 = s_1, \sigma_2 = s_1^2 - 2s_2, \sigma_3 = s_1^3 - 3s_1s_2 + 3s_3, \cdots$ etc. Any two solution systems for $\{x_i\}$ and $\{y_j\}$ must coincide up to permutations, so in general there must exist a permutation σ such that $x_i = y_{\sigma(i)}$. On the other hand all x_i and y_j are different or equal to zero; therefore equal to zero, which ends the proof. $\square$

It is interesting to note that because of the monotonicity property, computing the logical permanent of matrices with k-bounded arithmetic permanents, $perm(A) \le k$, for $k = 1, 2, \cdots$, does have superpolynomial $n^{\Omega(\log n)}$ monotone circuit complexity ([Ra 85]) for all k's. It stands in contrast with our

Corollary 1. The *Logical Permanent Problem* for matrices with k-bounded arithmetic permanents is computable within the uniform $O(\log^2 n)$ depth and $O(n^{4.5})$ size boolean circuits.

Theorem 2. If a bipartite graph G has a bounded permanent, $perm(A_G) \le k$ for k a constant, then the problem of constructing all matchings of G is in NC^2.

PROOF. One sees that the algorithm of Theorem 1 encodes matchings in the form of the numbers $\sum_i x_i^m - \sum_j y_j^m$ for $1 \le m \le cn^k$. The problem of decryption of these numbers and recovery of all actual matchings is a very interesting problem of polynomial algebra. We shall be able to prove the existence of such parallel 'matching recovery' in Lemma 3 for numbers of matchings bounded by $\log^{(\frac{1}{2}-\varepsilon)} n$.

However, we now apply for a constant k a completely different method which is an interesting new 'divide-and-conquer' approach to the problem of matching.

The following is the NC^2-algorithm for constructing all matchings of a given bipartite graph with a bounded permanent, $perm(A_G) \leq k$ for k a constant:

Input: Matrix A^ℓ

Subroutine (Split (A^ℓ)**).** Take in parallel all (i, j) entries of a matrix A such that $a_{ij} \neq 0$ and compute two new matrices $A_1^{\ell+1}$ and $A_2^{\ell+1}$:

- $A_1^{\ell+1}$ is the $(n-1) \times (n-1)$ matrix resulting from the cancellation of its i-th row and j-th column; store the numbers (i, j).

and

- $A_2^{\ell+1}$ is the $n \times n$ matrix resulting from plugging '0' into its (i, j)-entry.

Algorithm.

1. $A^0 \leftarrow A_G$
2. **Repeat in parallel**
   subroutine **Split** (A^ℓ)
   until $\ell = k - 1$.
3. Construct new matrices $N = (x_{ij})$ on the leaves of the computation tree. Suppose $M = (y_{ij})$ is on the leaf; then

$$x_{ij} = \begin{cases} p_{t_M(i,j)} & \text{if } y_{ij} = 1 \\ 0 & \text{otherwise} \end{cases}$$

4. Compute the determinants of all matrices N.
5. If a determinant is an encoding of a *unique matching* (the condition: $det(A_1^{k-1}) \neq 0$ <u>and</u> $det(A_2^{k-1}) = 0$ is fulfilled), recover it from the determinant (by consecutive dividing by prime numbers $p_1, p_2, \cdots, p_n$ and retrieving stored numbers (i, j) from the computational path) and print it out. (If you do not want repetition, do additional parallel sorting.) The correctness of the algorithm is based on the following

Lemma 2. For every matching in a graph G there exists a leaf of a computation tree (step 3) with the unique matching in it.

We now aim at improving Theorem 2. First we prove

Lemma 3. If a bipartite graph G has a permanent bounded by $\log^{\frac{1}{2}-\varepsilon} n$, $perm(A_G) \leq \log^{\frac{1}{2}-\varepsilon} n$, then the problem of constructing all its matchings lies in NC.

PROOF. Denote by k the number of matchings. Let $k < \log^{(\frac{1}{2}-\varepsilon)} n$, and $\{q_i\}$ are such primes that
1)$q_i > k$ (in fact $q_i \sim kn \log n < \log^{(1\frac{1}{2}-\varepsilon)} n$) and
2)$\prod_i q_i > 2^n(n!)^k > \sigma_j(x_1, \cdots, x_\ell), 1 \le j \le k,\ x_1, \cdots, x_k$ — products of primes plugged in matchings; the number of q_i is near kn.

Fix q_{i_0} (in parallel) and solve the system

$$\left.\begin{array}{ccc} \sum_{1\le i\le \ell} x_i - \sum_{1\le j\le k-\ell} y_j = A_1 \\ \vdots \quad\quad \vdots \quad\quad \vdots \\ \sum_{1\le i\le \ell} x_i^k - \sum_{1\le j\le k-\ell} y_j^k = A_k \end{array}\right. \left(\bmod q_{i_0}\right).$$

Take any solution $\bar{x}_1, \cdots, \bar{x}_\ell,\ \bar{y}_1, \cdots \bar{y}_{k-\ell}$ (at the beginning we test $\ell = 0, 1, \cdots, k$), then compute $\sigma_j(\bar{x}_1, \cdots, \bar{x}_\ell), \sigma_j(\bar{y}_1, \cdots, \bar{y}_{k-l}), 1 \le j \le k$. Any two solutions of this system coincide up to permutations in $\bar{x}_1, \cdots, \bar{x}_\ell$ and $\bar{y}_1, \cdots, \bar{y}_{k-\ell}$ (separately) because $q_{i_0} > k$. Therefore $\sigma_j(\bar{x}_1, \cdots, \bar{x}_\ell), \sigma_j(\bar{y}_1, \cdots, \bar{y}_{k-\ell})$ are *uniquely defined* and

$$\begin{aligned} \sigma_j(\bar{x}_1, \cdots, \bar{x}_\ell) &= \sigma_j(x_1, \cdots, x_\ell)(mod\, q_{i_0}) \\ \sigma_j(\bar{y}_1, \cdots, \bar{y}_{k-\ell}) &= \sigma_j(y_1, \cdots, y_{k-\ell})(mod\, q_{i_0}) \end{aligned}$$

where $x_1, \cdots, x_\ell, y_1, \cdots, y_{k-\ell}$ is the unique (up to permutations in $x_1, \cdots, x_\ell$ and in $y_1, \cdots, y_{k-\ell}$) solution of the system

$$\begin{cases} \sum x_i - \sum y_j = A_1 \\ \vdots \quad\quad \vdots \quad\quad \vdots \\ \sum x_i^k - \sum y_j^k = A_k \end{cases}$$

and so $\sigma_j(x_1, \cdots, x_\ell), \sigma_j(y_1, \cdots, y_{k-\ell})$ are defined uniquely.

By the Chinese remainder theorem restore $\sigma_j(x_1, \cdots, x_\ell),\ \sigma_j(y_1, \cdots, y_{k-\ell})$. It is possible since $\prod q_i > \sigma_j(x_1, \cdots, x_\ell),\ \sigma_j(y_1, \cdots, y_{k-\ell})$. Then apply [Lo 83] or [BKR 84] to find $x_1, \cdots, x_\ell,\ y_1, \cdots, y_{k-\ell}$.

The complexity of the solving system $mod\, q_{i_0}$ (the method of [ChG 83] and [ChG 84]) is polynomial in $(deg)^{(var)^2}$. $q_{i_0} \le k^{k^2} \cdot kn \log n$ is polynomial in n. The point is that the method of [ChG 83] and [ChG 84] can be done simultaneously in parallel time log (sequential time)$\le O(\log n)$ – its main subroutine is factoring in $\mathbb{F}_q[x_1, \cdots, x_n]$ – and the method in [ChG 83] needs only linear algebra – not reduction basis. $\square$

Having proved the existence of an NC-algorithm for the enumerator of $\log^{\frac{1}{2}-\varepsilon} n$ matchings (which seems to be a limit for an efficient parallel algebra algorithm), we are now going to attack the general matching problem of polynomially bounded permanents, both for the construction of a matching and the matching enumerator. We are able to prove a much stronger result than Lemma 2 by using our symmetric functions technique of Theorem 1 for the solution of the logical permanent problem (surprisingly not using any efficient linear algebra).

Theorem 3. If a bipartite graph G has a polynomial permanent ($perm(A_G) \leq cn^k$), then the problem of constructing a perfect matching lies in NC^3.

PROOF. Denote by $A = (a_{ij})$ a 0-1 $n \times n$ matrix. For any entry a_{ij}, by A_{ij} denote the $(n-1) \times (n-1)$ matrix obtained from A by canceling the i-th row and the j-th column.

For any $a_{ij} = 1$ test (with the help of the deciding method of Theorem 1) whether A_{ij} has at least one matching. We call such a_{ij} *generators*. Consider a row (i_0-th) containing at least two generators $a_{i_0 j_1} = a_{i_0 j_2} = 1$ (otherwise, if no such row exists, we have found a unique matching). Then at least one of the two matrices $A_{i_0 j_1}$ and $A_{i_0 j_2}$ has at most half of all the matchings of the matrix A. This is a crucial point of our algorithm (the rest is a consequence of our decision algorithm of Theorem 1)

Then apply the same construction to both matrices $A_{i_0 j_1}$, $A_{i_0 j_2}$ (call recursively the subroutine of Theorem 1), and so on. After $t \leq \log(cn^k) = O(\log n)$ steps we shall obtain one of the 2^t matrices with the unique matching. $\square$

Theorem 4. (Catching all Perfect Matchings in NC^3) If a bipartite graph G has a polynomial permanent ($perm(A_G) \leq cn^k$), then the problem of constructing all its perfect matchings lies in NC^3. The algorithm works in $O(\log^3 n)$ parallel time and $O(n^{3k+5.5} \log n)$ processors.

PROOF. We start with a definition:

Definition. A set of entries $a_{i_1 j_1}, \cdots, a_{i_u j_u}$ of the matrix A is called *(matching) active* if there exists a matching in the graph corresponding to the matrix A, containing all these entries.

One can test for any given set of entries $a_{i_1 j_1}, \cdots, a_{i_u j_u}$ whether it forms an *active set*. Namely, it is equivalent to the fact that for all $a_{i_1 j_1} = \cdots = a_{i_u j_u} = 1$, the indices $i_1, \cdots, i_u$ are pairwise distinct (and also $j_1, \cdots, j_u$) and besides, in the matrix $A^{(i_1, \cdots, i_u)}_{(j_1, \cdots, j_u)}$, obtained from A by canceling the rows $i_1, \cdots, i_u$ and the columns $j_1, \cdots, j_u$, there is at least one matching that can be checked by means of the decision procedure exposed above (Theorem 1).

Now we describe an algorithm yielding all the matchings of the matrix A. We can suppose w.l.o.g. that $n = 2^m$. The algorithm works recursively in $(m+1)$ stages. At the first stage it produces all the active entries.

Next, fix a certain i, $1 \le i \le m$, and assume that after the i-th stage the algorithm has produced the family of all the active sets of entries of the form $a_{2^{(i-1)}\cdot s+1, j_1}, \cdots, a_{2^{(i-1)}\cdot s+2^{(i-1)}, j_{2^{(i-1)}}}$ for each s, $0 \le s < 2^{m-i+1}$.

So, at the $(i+1)$-th stage for every $0 \le t < 2^{m-i}$ the algorithm tests in parallel for any pair of active sets of the form $a_{2^{(i-1)}(2t)+1, j_1}, \cdots, a_{2^{(i-1)}(2t)+2^{(i-1)}, j_{2^{(i-1)}}}$ and $a_{2^{(i-1)}(2t+1)+1, p_1}, \cdots, a_{2^{(i-1)}(2t+1)+2^{(i-1)} p_{2^{(i-1)}}}$ whether the union of these two sets $a_{2^{(i-1)}(2t)+1, j_1}, \cdots, a_{2^{(i-1)}(2t+1)+2^{(i-1)}, p_{2^{(i-1)}}}$ forms an active set. If yes, then the algorithm outputs it as one of the results of the $(i+1)$-th stage. This completes the description of the algorithm. At the end of it (after $(m+1)$ stages) we obtain all the matchings of the matrix A.

Let us prove that the described algorithm is in NC. The depth of the algorithm is $O(\log^3 n)$. To estimate the size of the algorithm observe that after any stage there would be less than $n \cdot cn^k$ active sets. Thus, at any stage the algorithm tests less than $c^2 n^{2k+1}$ pairs of active sets. This proves that the described algorithm lies in NC^3 and takes $O(n^{3k+5.5} \log n)$ processors. $\square$

We now derive some important corollaries from the construction of Theorems 3 and 4:

Corollary 2. The problem of computing a polynomially bounded permanent is in NC^3.

Corollary 3. If the number of matchings in a graph G is $n^{O(\log n)}$, then the decision problem (logical permanent) and the construction of a perfect matching problem are mutually $O(\log^2 n)$-uniform depth reducible.

Corollary 4. If a bipartite graph G has a permanent less than $2^{\log^k n}$, then there is a $\log^{k+1} n$ parallel time ($\log^{k+1} n$-sequential space) algorithm for enumerating all perfect matchings.

Corollary 5. If a bipartite graph G has a permanent less than 2^{n^ε} for a constant $\varepsilon < 1$, then there is a sublinear parallel time (sublinear sequential space) algorithm for enumerating all the perfect matchings in a graph.

3. 'Optimal' Parallel Time Enumerator Algorithm.

We consider now the computational problem of enumerating all the perfect matchings in an arbitrary bipartite graph. A lower bound for the parallel (boolean) time is $\Omega(\log(perm(A)))$ for $perm(A)$, say, at least linear, $perm(A) \geq n$ (the worst case is $\Omega(n \log n)$). We are now interested in the best possible parallel enumerators for (big sized) permanents not covered by Theorem 4. The enumerator algorithm of Theorem 4 can be reused now to design the 'optimal' up to the $\log n$-factor parallel time enumerator algorithm:

Theorem 5. There exists an $O(\log(perm(A)) + \log^2 n)\log n$ parallel time (-uniform boolean depth) algorithm for enumerating all the perfect matchings in an arbitrary bipartite graph.

PROOF. Given an arbitrary bipartite graph G with the adjacency matrix A. The parallel algorithm for the logical permanent of A (Theorem 1) can be designed working in $O(\log(perm(A)) + \log^2 n)$ parallel time. Now we generalize the enumerator algorithm of Theorem 4 for the case of graphs with arbitrary permanents. We reduce the resulting unbounded fan-in at every stage to the bounded one on the expense of $O(\log(perm(A)))$-depth. This yields an algorithm working in $O(\log(perm(A)) + \log^2 n)\log n$-parallel time $\square$

Corollaries 4 and 5 are now special cases of Theorem 5.

4. Deciding whether the Permanent is Small; a Randomised Version of the Matching Enumerator.

It is known that for every positive integer k there exists a $(0,1)$-matrix with the permanent k. The minimum order of $(0,1)$-matrices with the permanent k does not exceed $\lceil \log(k-1) \rceil + 2$ for $k = 2, 3, \cdots$ ([GMW 74]). An important computational problem of *bounded counting* arises: given an arbitrary k, $k = 0, 1, 2, 3, \cdots$, decide whether $perm(A)$ is k-*small*, i.e. whether $perm(A) \leq k$. If the answer is **yes**, our enumerator algorithm of Section 2 will produce all the perfect matchings. Our algorithms provide a way of deciding whether $perm(A) = k$, for $k > 0$, but unfortunately they cannot distinguish between *zero* and *many* matchings.

A similar situation holds for polynomially *small* permanents. For a function $f \in n^{O(1)}$, $perm(A)$ is f-*small* if $perm(A) \leq f(n)$ for an $n \times n$-matrix A. We are now interested in detecting all matrices A with f-small permanents. We produce here an attractive randomized version of our Theorem 4.

Theorem 6. (Randomised Enumerator) For any polynomial $f \in n^{O(1)}$ $(f(n) =$

cn^k) there exists a randomized (Las Vegas) RNC^3-algorithm for deciding whether $perm(A)$ is f-small. In the case $perm(A)$ is f-small, the algorithm outputs all the perfect matchings of A. The algorithm takes $O(\log^3 n)$ parallel time and $O(n^{2k+6.5} \log n)$ processors.

PROOF. There exists a Las Vegas RNC^2-algorithm (not outputting any errors) for the logical permanents (cf. [MVV 87], [KUW 85],[Ka 86]) working in $O(\log^2 n)$ parallel time and $O(n^{5.5})$ processors and using $O(n^2 \log n)$ random bits. We use this algorithm (instead of applying the deterministic procedure of Theorem 4) to compute the logical permanent of the *active set* matrices $A^{i_1,\cdots,i_u}_{j_1,\cdots,j_u}$ in the algorithm of Theorem 4.

We control the number of active sets produced at any level by comparing it in parallel with the number $n \cdot cn^k$ (computed by another NC^1-circuit). If it exceeds this number, we switch the circuit off. If not, we shall obtain a printout of all the matchings in A in $O(\log^3 n)$ parallel time. The algorithm takes $O(n^{2k+6.5} \log n)$ processors. $\square$

The randomized enumerator algorithm above reduces the number of processors by the factor of $\sim O(n^k)$ on the expense of $O(n^{2k+8.5} \log^2 n)$ random bits.

Remark. As an immediate application of the randomized enumerator algorithm, we observe that the problem of checking whether $perm(A) = det(A)$ for any given $0-1$ matrix A with $det(A) = n^{O(1)}$ has been put in RNC^3. It is also interesting to note that the general problem of testing whether $perm(A) = det(A)$ ([VY 87]) for $0-1$ matrices is polynomial time equivalent to the problem of checking whether a given bipartite graph has a Pfaffian orientation ([LP 86]), and to the Even Cycle Problem ([VY 87]) for directed graphs.

5. Extensions.

Our results can be extended to the problem of Maximum Matching for the case of non-bipartite graphs with the polynomially bounded number of matchings. In this case we deal with computations over skew matrices and Pfaffian functions rather than bipartite adjacency matrices. Due to the enumerator algorithm of Theorem 4, the problems of Maximum Weighted Matching (with weights in binary), Exact Matching (cf. [MVV 87]), First Lexicographical Perfect Matching, or the connected Stable Marriage Problems are all put in NC^3, provided the number of underlying matchings is small. Also, as a consequence of the enumerator, inherently difficult problems of counting $perm(A)$ mod k ([Va 79], [VV 85]) have been proved efficiently parallelisable for the polynomially small permanents.

6. Further Research.

It remains to be seen whether the method applied in our algorithm for bounded cases of the logical permanent could be refined to provide a general deterministic solution. It seems that a more careful look at the *algebraic varieties* stemming from our *symmetric functions* construction of Theorem 1 is now justified.

Independently, it would be very nice to shed some light (say, via NC-reducibilities) on the mutual interdependence between the decision methods and the construction of a perfect matching for graphs with superpolynomial permanents (Theorem 3 and Corollary 3 might be good starting points).

Acknowledgements.

We are thankful to Avi Wigderson, Volker Strassen and Mark Jerrum for a number of interesting conversations. Special thanks also go to Erich Kaltofen for valuable comments concerning parallel algebra for computing integer roots of polynomials used in Lemma 3. Finally we thank Michael Ben-Or, Noga Alon and Michael Rabin for commenting on the preliminary draft of this paper.

References.

[BKR 84] Ben-Or, M., Kozen, D., and Reif, J., *The Complexity of Elementary Algebra and Geometry*, *Proc.* 16^{th} ACM STOC (1984), pp.457-464

[BCP 83] Borodin, A., Cook, S.A., and Pippenger, N., *Parallel Computation for Well-Endowed Rings and Space-Bounded Probabilistic Machines*, Information and Control **58** (1983), pp.113-136

[BGH 82] Borodin, A., von zur Gathen, J., and Hopcroft, J., *Fast Parallel Matrix and GCD Computation*, Proc. 18^{th} IEEE FOCS (1982), pp. 65-71

[Br 86] Broder, A.Z., *How Hard is it to Marry at Random (On the Approximation of the Permanent)*, Proc. 18^{th} ACM STOC (1986), pp. 50-58

[ChG 83] Chistov, A.L., and Grigoriev, D.Yu., *Subexponential-Time Solving Systems of Algebraic Equations I/II*, LOMI Reports E-9-83, E-10-83, Steklov Mathematical Institute (1983), pp. 1-57, pp. 1-62

[ChG 84] Chistov, A.L., and Grigoriev, D.Yu., *Fast Decomposition of Polynomials into Irreducible Ones and the Solution of Systems of Algebraic Equations*, Soviet Math. Dokl. (1984), pp. 380-383

[Co 85] Cook, S.A., *A Taxonomy of Problems with Fast Parallel Algorithms*, Information and Control **64** (1985), pp. 2-22

[DK 86a] Dahlhaus, E., and Karpinski, M., *The Matching Problem for Strongly Chordal Graphs is in NC*, Research Report No. 855-CS, University of Bonn (1986), pp. 1-8

[DK 86b] Dahlhaus, E. and Karpinski, M., *Perfect Matching for Regular Graphs is AC^0-Hard for the General Matching Problem*, Research Report No. 858-CS, University of Bonn (1986), pp. 1-5

[Fa 83] Farber, M., *Characterizations of Strongly Chordal Graphs*, Discrete Math. **43** (1983), pp. 173-189

[FLS 85] Furst, M., Lipton, R., Stockmeyer, L., *Pseudorandom Number Generation and Space Complexity*, Information and Control **64** (1985), pp. 43-51

[FSS 81] Furst, M., Saxe, J.B., and Sipser, M., *Parity, Circuits, and the Polynomial Time Hierarchy*, Proc. 22^{nd} IEEE FOCS (1981), pp. 260-270

[Ga 60] Gantmacher, F.R., *The Theory of Matrices*, Chelsea Publ., New York (1960), pp. 1-374

[vzG 86] von zur Gathen, J., *Permanent and Determinant*, Proc. 22^{nd} IEEE FOCS (1986), pp.398-401

[GT 86] Goldberg, A.V., and Tarjan, R.E., *A New Approach to the Maximum Flow Problem*, Proc. 18^{th} ACM STOC (1986), pp.136-146

[GMW 74] Gordon, B., Motzkin, T.S., and Welch, L., *Permanents of* $0,1$-*matrices*, J. Computational Theory Ser. A **17** (1974), pp. 145-151

[HM 86] Hemboldt, D., and Mayr, E., *Two Processor Scheduling is in NC*, Proc. VLSI Algorithms and Architectures, LNCS **227** (1986), pp. 12-25

[HK 73] Hopcroft, J.E., and Karp, R.M., *An* $n^{\frac{5}{2}}$ *Algorithm for Maximum Matching in Bipartite Graphs*, SIAM J. Comp. **2** (1973), pp. 225-231

[Ka 86] Karloff, H.J., *A Las Vegas* RNC *Algorithm for Maximum Matching*, Combinatorica **6** (1986), pp. 387-391

[KUW 85] Karp, R.M., Upfal, E., and Wigderson, A., *Constructing a Perfect Matching is in Random NC*, Proc. 17^{th} ACM STOC (1985), pp. 22-32

[KVV 85] Kozen, D., Vazirani, U.V., and Vazirani, V.V., *NC Algorithms for Comparability Graphs, Interval Graphs, and Testing for Unique Perfect Matching*, Proc 5^{th} Conference on Foundations of Software Technology and Theoretical Computer Science, Springer LNCS Vol. 206 (1985), pp. 496-503

[LPV 81] Lev, G., Pippenger, N., and Valiant, L., *A Fast Parallel Algorithm for Routing in Permutation Networks*, IEEE Transactions on Computers Vol. **C-30** (1981), pp. 93-100

[Lo 83] Loos, R., *Computing Rational Zeros of Integer Polynomials by p-adic Expansions*, SIAM J.Comp.**12** (1983), pp.286-293

[LP 86] Lovász, L., and Plummer, M.D., *Matching Theory*, North Holland Mathematical Studies Vol 29 (1986), pp. 1-544

[Ma 79] Macdonald, I.G., *Symmetric Functions and Hall Polynomials*, Clarendon Press, Oxford (1979), pp.1-180

[MVV 87] Mulmuley, K., Vazirani, U.V., and Vazirani, V.V., *Matching is as Easy as Matrix Inversion*, Proc. 19^{th} ACM STOC (1987), pp. 345-354

[NW 75] Nijenhuis, A., and Wilf, H., *Combinational Algorithms*, Academic Press, (1975), pp.1-253

[RV 84] Rabin, M.D., and Vazirani, V.V., *Maximum Matchings in General Graphs through Randomisation*, TR-15-84, Harvard University Center for Research in Computing Technology (1984)

[Ra 85] Razborov, A.A., *Bound on the Monotone Network Complexity of the Logical Permanent*, Matem. Zametki **37** (1985), in Russian

[Sm 87] Smolensky, R., *Algebraic Methods in the Theory of Lower Bounds for Boolean Circuit Complexity*, Proc. 19^{th} ACM STOC (1987), pp. 77-82

[St 83] Stockmeyer, L., *The Complexity of Approximate Counting*, Proc. 15^{th} ACM STOC (1983), pp. 118-126

[Ta 85] Tarjan, R.E., *Decomposition by Clique Separators*, Discrete Mathematics **55** (1985), pp. 221-232

[Va 79] Valiant, L.G., *The Complexity of Computing the Permanent*, Theoretical Computer Science **8** (1979), pp. 189-201

[VV 85] Valiant, L.G., and Vazirani, V.V., *NP is as Easy as Detecting Unique Solutions*, Proc. 17^{th} ACM STOC (1985), pp. 458-463

[VY 87] Vazirani, V.V., and Yannakakis, M., *Pfaffian Orientations,* 0/1 *Permanents, and Even Cycles in Directed Graphs*, manuscript (1987)