On the Bipartite Unique Perfect Matching Problem

Thanh Minh Hoang*[1], Meena Mahajan[2], and Thomas Thierauf[3]

[1] Abt. Theor. Inform., Universität Ulm, 89069 Ulm, Germany
[2] Inst. of Math. Sciences, Chennai 600 113, India
[3] Fak. Elektr. und Inform., Aalen University, 73430 Aalen, Germany

Abstract. In this note, we give tighter bounds on the complexity of the bipartite unique perfect matching problem, bipartite-UPM. We show that the problem is in $\mathbf{C_{=}L}$ and in $\mathbf{NL}^{\oplus\mathbf{L}}$, both subclasses of $\mathbf{NC}^2$.
We also consider the (unary) weighted version of the problem. We show that testing uniqueness of the minimum-weight perfect matching problem for bipartite graphs is in $\mathbf{L}^{\mathbf{C_{=}L}}$ and in $\mathbf{NL}^{\oplus\mathbf{L}}$.
Furthermore, we show that bipartite-UPM is hard for **NL**.

1 Introduction

The perfect matching problem PM asks whether there exists a perfect matching in a given graph. PM was shown to be in **P** by Edmonds [5], but it is still open whether there is an **NC**-algorithm for PM. In fact, PM remains one of the most prominent open questions in complexity theory regarding parallelizability. It is known to be in randomized **NC** (**RNC**) by Lovász [12]; subsequently Karp, Upfal and Wigderson [9], and then Mulmuley, Vazirani, and Vazirani [13] showed that constructing a perfect matching, if one exists, is in $\mathbf{RNC}^3$ and $\mathbf{RNC}^2$, respectively. Recently, Allender, Reinhardt, and Zhou [3] showed that PM (both decision and construction) is in non-uniform **SPL**. However, to date no **NC** algorithm is known for PM.

In this paper we consider the complexity of the unique perfect matching problem posed by Lovász (see [10]), UPM for short. That is, for a given graph G, one has to decide whether there is precisely one perfect matching in G. Furthermore, we consider the problem of testing if a (unary) weighted graph has a unique minimum-weight perfect matching. The latter problem has applications in computational biology. The unique maximum-weight perfect matching can be used to predict the folding structure of RNA molecules (see [17]).

Gabow, Kaplan, and Tarjan [6] observed that UPM is in **P**. Kozen, Vazirani, and Vazirani [10, 11] showed that UPM for bipartite graphs is in **NC**. Their techniques don't seem to generalize to arbitrary graphs (see Section 3.2 for more detail). [4]

* Supported by DFG grant Scho 302/7-1.

[4] In [10] it is claimed that the technique works also for the general case, but this was later retracted in a personal communication by the authors; see also [11].

In this paper we give tighter bounds on the complexity of UPM for bipartite graphs (bipartite-UPM, for short). Our bounds place bipartite-UPM into complexity classes lying between logspace $\mathbf{L}$ and $\mathbf{NC}^2$. The classes we consider are non-deterministic logspace ($\mathbf{NL}$), exact counting in logspace ($\mathbf{C_{=}L}$), and logspace counting modulo 2 ($\oplus\mathbf{L}$). Some known relationships among these classes and their relativized versions are as follows:

$$\mathbf{L} \subseteq \mathbf{NL} \subseteq \mathbf{C_{=}L} \subseteq \mathbf{L}^{\mathbf{C_{=}L}} = \mathbf{NL}^{\mathbf{C_{=}L}} \subseteq \mathbf{NC}^2, \qquad \mathbf{L} \subseteq \oplus\mathbf{L} \subseteq \mathbf{NL}^{\oplus\mathbf{L}} \subseteq \mathbf{NC}^2.$$

These classes are important because they capture, via completeness, the complexities of important naturally defined problems. Reachability in directed graphs is complete for $\mathbf{NL}$, as also 2-CNF-SAT. Testing whether a square matrix over integers is singular is complete for $\mathbf{C_{=}L}$, and computing the rank of an integer matrix is complete for $\mathbf{L}^{\mathbf{C_{=}L}}$. A complete problem for $\oplus\mathbf{L}$ is deciding whether the number of perfect matchings in a bipartite graph is odd.

Our results (from Section 3) place bipartite-UPM in $\mathbf{C_{=}L} \cap \mathbf{NL}^{\oplus\mathbf{L}}$. The first upper bound implies that G is in bipartite-UPM if and only if an associated matrix A, obtainable from G via very simple reductions (projections), is singular. We show in Section 4 that (unary) weighted bipartite-UPM is in $\mathbf{L}^{\mathbf{C_{=}L}} \cap \mathbf{NL}^{\oplus\mathbf{L}}$. By the preceding upper bounds, it might well be the case that bipartite-UPM is easier than the perfect matching problem. However, we show in Section 5 that bipartite-UPM is hard for $\mathbf{NL}$; thus the best known lower bounds for PM and for UPM coincide. Our results thus place bipartite-UPM between $\mathbf{NL}$ and $\mathbf{C_{=}L}$. Furthermore, our results provide a new complete problem for $\mathbf{NL}$. This is the problem of testing if a given perfect matching is unique in a bipartite graph.

2 Preliminaries

Complexity Classes: $\mathbf{L}$ and $\mathbf{NL}$ denote languages accepted by deterministic and nondeterministic logspace bounded Turing machines, respectively. For a nondeterministic Turing machine M, we denote the number of accepting and rejecting computation paths on input x by $acc_M(x)$ and by $rej_M(x)$, respectively. The difference of these two quantities is gap_M, i.e., for all x: $gap_M(x) = acc_M(x) - rej_M(x)$. The complexity class $\mathbf{GapL}$ is defined as the set of all functions $gap_M(x)$ where M is a nondeterministic logspace bounded Turing machine. The class $\mathbf{C_{=}L}$ (*Exact Counting in Logspace*) is the class of sets A for which there exists a function $f \in \mathbf{GapL}$ such that $\forall\ x : \quad x \in A \iff f(x) = 0$. $\mathbf{C_{=}L}$ is closed under union and intersection, but is not known to be closed under complement. $\oplus\mathbf{L}$ is the class of sets A for which there exists a function $f \in \mathbf{GapL}$ such that $\forall\ x : \quad x \in A \iff f(x) \equiv 0 \pmod 2$. $\oplus\mathbf{L}$ is closed under Turing reductions. Circuit classes $\mathbf{NC}^k$ are all families of languages or functions that can be computed by polynomial-size circuits of depth $O(\log^k n)$.

Perfect Matchings: Let $G = (V, E)$ be an undirected graph. A *matching* in G is a set $M \subseteq E$ such that no two edges in M have a vertex in common. A matching M is called *perfect* if every vertex from V occurs as the endpoint of some edge in M. By $\#\ \mathrm{pm}(G)$ we denote the number of perfect matchings in G.

The perfect matching problem and the unique perfect matching problem are defined as $\mathrm{PM} = \{\, G \mid \#\ \mathrm{pm}(G) > 0 \,\}$ and $\mathrm{UPM} = \{\, G \mid \#\ \mathrm{pm}(G) = 1 \,\}$. Restricted to bipartite graphs, we denote the problem bipartite-UPM. We also consider the problem of testing whether there exists precisely one perfect matching with minimal weight in a weighted graph.

For graph G with n vertices, the (order n) *skew-symmetric adjacency matrix* A is as defined below:

$$a_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E \text{ and } i < j, \\ -1 & \text{if } (i,j) \in E \text{ and } i > j, \\ 0 & \text{otherwise.} \end{cases}$$

By transforming $a_{i,j} \quad \mapsto \quad a_{i,j}(x) = a_{i,j}x_{i,j}$, for indeterminate $x_{i,j} = x_{j,i}$, we get a skew-symmetric variable matrix $A(x)$ called the *Tutte's matrix* of G.

Theorem 1 (Tutte 1952). $G \in \mathrm{PM} \iff \det(A(x)) \neq \mathbf{0}$.

Since $\det(A(X))$ is a symbolic multivariate polynomial, it can have exponential length in n, when written as a sum of monomials. However, there are randomized identity tests for polynomials that just need to evaluate a polynomial at a random point [18, 15]. Since the determinant of an integer matrix is complete for **GapL**, a subclass of $\mathbf{NC}^2$, Lovász observed that Tutte's Theorem puts PM in $\mathbf{RNC}^2$.

It is well known from linear algebra that, for an $n \times n$ skew-symmetric matrix ($A = -A^T$), $\det(A) = 0$ if n is odd and $\det(A) = \det(A^T) \geq 0$ if n is even.

The following fact is a consequence of Tutte's Theorem:

Fact 1 *1.* $\#\ \mathrm{pm}(G) = 0 \implies \det(A) = 0$,
2. $G \in \mathrm{UPM} \implies \det(A) = 1$.

Rabin and Vazirani [14] used Fact 1 for reconstructing the unique perfect matching as follows. Let G be in UPM with the unique perfect matching M. Let $G_{i,j} = G - \{i,j\}$ denote the subgraph of G obtained by deleting vertices i and j, and let $A_{i,j}$ be the skew-symmetric adjacency matrix of $G_{i,j}$. For each edge $(i,j) \in E$, one can decide whether (i,j) belongs to M or not by:

$$\begin{array}{lllll} (i,j) \in M & \implies & G_{i,j} \in \mathrm{UPM} & \implies & \det(A_{i,j}) = 1, \\ (i,j) \notin M & \implies & G_{i,j} \notin \mathrm{PM} & \implies & \det(A_{i,j}) = 0. \end{array}$$

Hence, if $G \in \mathrm{UPM}$ we can compute the perfect matching by looking at the values $\det(A_{i,j})$ for all edges (i,j) of G.

3 Testing unique perfect matching

3.1 Bipartite UPM is in $\mathbf{L}^{\mathbf{C_=L}} \cap \mathbf{NL}^{\oplus\mathbf{L}}$

As seen in the last section, if $G \in \mathrm{UPM}$ then the unique perfect matching M in G can be easily computed [14]. Our approach is to assume $G \in \mathrm{UPM}$ and attempt to construct some perfect matching M as above. If this succeeds, then we check whether M is unique.

Note that any perfect matching can be represented as a symmetric permutation matrix. We construct the matrix $B = (b_{i,j})$ of order n, where

$$b_{i,j} = |a_{i,j}| \det(A_{i,j}).$$

Since A and each $A_{i,j}$ are skew-symmetric, B is symmetric non-negative. From the discussion above we have

Lemma 1. *$G \in \mathrm{UPM} \implies B$ is a symmetric permutation matrix.*

The first step of our algorithm for UPM is to check that the symmetric matrix B is indeed a permutation matrix. This is so if and only if every row contains precisely one 1 and all other entries are 0. This is equivalent to

$$\sum_{i=1}^{n} \left(\left(\sum_{j=1}^{n} b_{ij} \right) - 1 \right)^2 = 0. \qquad (1)$$

Since all b_{ij}'s can be computed in **GapL**, the expression on the left hand side in equation (1) can be computed in **GapL** too. We conclude

Lemma 2. *$\{\, G \mid B$ is a permutation matrix$\} \in \mathbf{C_{=}L}$.*

Now assume that B is a permutation matrix (if not, already $G \notin \mathrm{UPM}$), and therefore defines a perfect matching M in G. Suppose there is another perfect matching M' in G. Then the graph $(V, M \triangle M')$ is a union of disjoint alternating cycles, defined below.

Definition 1. *Let M be a perfect matching in G. An* alternating cycle *in G* with respect to *M is an even simple cycle that has alternate edges in M and not in M.*

Lemma 3. *Let M be a perfect matching in G. $G \in \mathrm{UPM}$ if and only if G has* no *alternating cycle with respect to M.*

```
ALTERNATING CYCLE($G, B$)
1  guess $s \in V$
2  $i \leftarrow s$
3  repeat
4      guess $j \in V$
5      if $b_{i,j} = 0$ then reject
6      guess $k \in V \setminus \{i\}$
7      if $a_{j,k} = 0$ then reject
8      $i \leftarrow k$
9  until $k = s$
10 accept
```

Given graph G and a permutation matrix B that defines some perfect matching M in G, algorithm Alternating Cycle searches for an alternating cycle in G with respect to M in nondeterministic logspace. It guesses a node s of an alternating cycle in G (line 1). Assume that we are at node i in the moment. Then Alternating Cycle makes two steps away from i. The first step is to node j such that $(i, j) \in M$ (line 4 and 5), the second step is to a neighbor $k \neq i$ of j such that $(j,k) \notin M$ (line 6 and 7). If $k = s$ in line 9 we closed a cycle of length at least 4 that has edges alternating in M and not in M. Note that the cycle may not be simple. However, if G is bipartite, then all cycles in G have even length, and so we have visited at least one alternating cycle on the way. Since **NL** equals co-**NL** [8, 16], we have

Lemma 4. *Given a bipartite graph G and a matching M in G, testing if $G \in$* UPM *is in* **NL**.

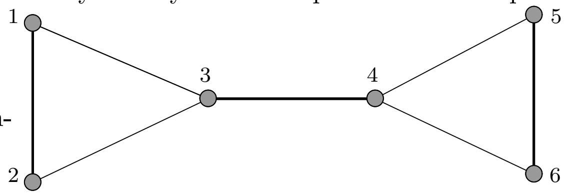
Consider the setting that Alternating Cycle has just graph G as input, and matrix B is provided by the oracle set $S = \{ (G,i,j,c) \mid b_{i,j} = c \} \in \mathbf{C_{=}L}$. Then Alternating Cycle can be implemented in $\mathbf{NL}^S$. Note that the oracle access is very simple: the queries are just a copy of the input and of some variables. In particular, this fulfills the Ruzzo-Simon-Tompa restrictions for oracle access by space bounded Turing machines. Since $\mathbf{NL}^{\mathbf{C_{=}L}} = \mathbf{L}^{\mathbf{C_{=}L}}$ [2], we have

Lemma 5. *Let G be a bipartite graph such that B is a permutation matrix. Then in $\mathbf{L}^{\mathbf{C_{=}L}}$ we can decide whether G is in* UPM.

Combining the algorithms from Lemma 2 and 5, we obtain the following:

Theorem 2. bipartite-UPM $\in \mathbf{L}^{\mathbf{C_{=}L}}$.

Unfortunately, Alternating Cycle works correctly only for bipartite graphs, since these do not have any odd cycles. On input of an non-bipartite graph, Alternating Cycle might accept a cycle of alternating edges with respect to some perfect matching which is *not* simple, thereby possibly giving a false answer. The graph G alongside provides an example. The unique perfect matching is $M = \{(1,2),(3,4),(5,6)\}$, but Alternating Cycle outputs 'accept'.



```
Permutation($B$)
1  for $i \leftarrow 1$ to $n$ do
2      $k \leftarrow 0$; $l \leftarrow 0$
3      for $j \leftarrow 1$ to $n$ do
4          if $b_{i,j} \equiv 1 \pmod 2$
               then $k \leftarrow k+1$
5          if $b_{j,i} \equiv 1 \pmod 2$
               then $l \leftarrow l+1$
6      if $k \neq 1$ or $l \neq 1$
           then reject
7  accept
```

Interestingly, we can also obtain $\mathbf{NL}^{\oplus\mathbf{L}}$ as an upper bound for bipartite-UPM. Recall the oracle set S we use in the above algorithm. In all the queries we have $c = 0$ or $c = 1$. Suppose we replace S by the set $T \in \oplus\mathbf{L}$,

$$T = \{ (G,i,j,c) \mid b_{i,j} \equiv c \pmod 2 \}.$$

It is easy to design a deterministic logspace algorithm, see Permutation alongside, that, with oracle access to T, checks whether B is a permutation matrix over $\mathbf{Z}_2$.

Since $\mathbf{L}^{\oplus\mathbf{L}} = \oplus\mathbf{L}$, we have

Lemma 6. *$\{G \mid B$ is a permutation matrix$\} \in \oplus\mathbf{L}$.*

Consider algorithm Alternating Cycle with oracle T. Although we might get different oracle answers when switching from S to T, it is not hard to check that we anyway get the correct final answer. Again we combine the two steps and get

Corollary 1. bipartite-UPM $\in \mathbf{NL}^{\oplus\mathbf{L}}$.

3.2 Bipartite-UPM is in $\mathbf{C_{=}L}$

Based on the method in [11], the upper bound $\mathbf{L}^{\mathbf{C_{=}L}}$ for bipartite UPM can be improved to $\mathbf{C_{=}L}$. Note that we do not know whether $\mathbf{L}^{\mathbf{C_{=}L}} = \mathbf{C_{=}L}$.

Let $G = (U, V, E)$ be a bipartite graph with $|U| = |V| = n$. Let A be the bipartite adjacency matrix of G; A is of order n. Then the skew-symmetric adjacency matrix of G is of the form $S = \begin{pmatrix} \mathbf{0} & A \\ -A^T & \mathbf{0} \end{pmatrix}$. Since $\det(S) = \det^2(A)$, Fact 1 gives the following for bipartite graph G.

Fact 2 *1.* $\#\,\mathrm{pm}(G) = 0 \implies \det(A) = 0$,
2. $G \in \mathrm{UPM} \implies \det(A) = \pm 1$.

The following lemma puts the idea of Kozen, Vazirani, and Vazirani [10] in such a way that we get $\mathbf{C_{=}L}$ as an upper bound.

Lemma 7. *For bipartite graph G with $2n$ vertices and bipartite adjacency matrix A, define matrices $B = (b_{i,j})$ and C of order n as follows*

$$\begin{aligned} b_{i,j} &= a_{i,j}\ \det^2(A_{i|j}),\ \text{for } 1 \le i, j \le n, \\ C &= I - AB^T, \end{aligned}$$

where I is the $n \times n$ identity matrix and $A_{i|j}$ is the sub-matrix obtained by deleting the i-th row and the j-th column of A. Then G has a unique perfect matching if and only if

(i) B is a permutation matrix, and
(ii) the characteristic polynomial of C is $\chi_C(x) = x^n$.

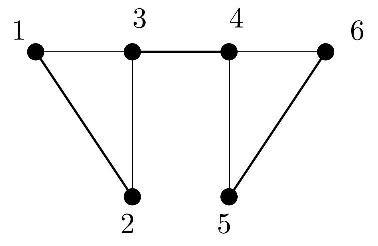
We provide some intuition to the lemma. Just as in the general case of Lemma 2, if B is a permutation matrix, then matrix B describes a perfect matching. The product AB^T puts the matching edges on the main diagonal of the matrix. Then $I - AB^T$ takes out the matching edges. Now consider C as the adjacency matrix of a (directed) graph, say H. This can be thought of as identifying vertex i from the left-hand side with vertex i from the right-hand side of the bipartite graph AB^T. (i.e. if G has vertices $U = \{u_1, \ldots, u_n\}$ and $W = \{w_1, \ldots, w_n\}$, then AB^T matches u_i with w_i, and edge (i, j) in H corresponds to path (u_i, w_i, u_j) in G.) Then any cycle in graph H corresponds to an alternating cycle in G. Hence there should be no cycles in H. Equivalently, all coefficients of the characteristic polynomial of C should be 0.

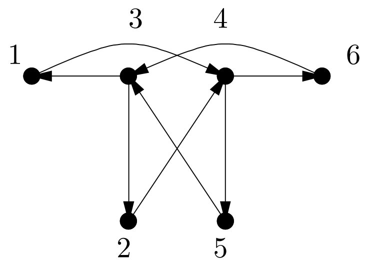
Another way of seeing this is as follows: $\chi_C(x) = \det(xI - C) = \det((x-1)I + AB^T)$. But $(x-1)I + AB^T$ is the bipartite adjacency matrix of G when vertices are renumbered to get the matching edges (of B) on the main diagonal, and with weights x on these matched edges, weights 1 on other edges. So condition (ii) checks if the determinant of this matrix is x^n.

We consider the complexity of checking the conditions of Lemma 7. Regarding the condition (i), the problem of testing if B is a permutation matrix is essentially the same as in the general case and can be done in $\mathbf{C_=L}$ (Lemma 2). Consider the condition (ii). Here the elements of matrix C are not given as input, they are certain determinants. However, a result in [1] shows that composition of determinants is computable again in **GapL**, i.e. the coefficients of the characteristic polynomial of C can be computed in **GapL** and they can be verified in $\mathbf{C_=L}$ [7]. Therefore condition (ii) can be checked in $\mathbf{C_=L}$. We conclude:

Theorem 3. bipartite-UPM $\in \mathbf{C_=L}$.

The above technique doesn't seem to generalize to non-bipartite graphs. The graph G shown here provides an example where the technique doesn't seem to work. Observe that $G \in$ UPM with the unique perfect matching $M = \{(1,2),(3,4),(5,6)\}$. Permute G and take out the edges of M as described above. This leads to graph H. Since $G \in$ UPM, H should have no directed cycles. But H contains 4 directed cycles, one of them is $(1,4,5,3)$. Another problem comes from the sign of the cycles in H.

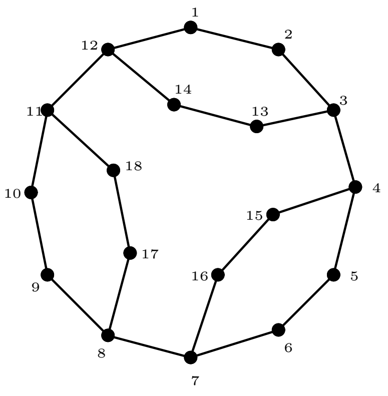




G H

Considering the concept of the skew-symmetric matrix of a non-bipartite graph, we ask whether there is an analog to Lemma 7. Namely, let A be the skew-symmetric adjacency matrix of G, and let B be defined as: $b_{i,j} = a_{i,j}\det(A_{i,j|i,j})$. ($A_{i,j|i,j}$ is the skew-symmetric adjacency matrix of the graph $G_{i,j|i,j}$ obtained by deleting vertices i and j, so we delete these rows and columns from A.) Then an analogous test for whether $G \in$ UPM would be: Does B correspond to some perfect matching M, and does weighting the edges of M with x give matrix A_x with determinant x^n? Unfortunately, this is not true.

The graph G shown alongside provides an example (even in the bipartite case). Obviously, G is not in UPM. But the matrix $B = (b_{i,j})$ computed by the expression $b_{i,j} = a_{i,j}\det(A_{i,j|i,j})$ is a symmetric permutation matrix which corresponds to the perfect matching $M = \{(1,2),(3,4),(5,6),(7,8),(9,10),(11,12),(13,14),(15,16),(17,18)\}$. Let B' be the skew-symmetric adjacency matrix corresponds to M. By Maple we get $\chi_C(x) = x^{18}$ where $C = I - AB'^T$.



4 Unique minimum-weight perfect matching

We now consider graphs with positive edge-weights. We assume that the weights are given in unary (or are bounded by a polynomial in the number of vertices). It has been shown by Mulmuley, Vazirani, and Vazirani [13] that there is a $\mathbf{RNC}^2$ algorithm (by using Isolating Lemma) for computing some perfect matching in graph G. This algorithm chooses random weights for the edges of G, then the Isolating Lemma states that with high probability, there is a unique perfect matching of minimal weight. We describe briefly the procedure by [13] for reconstructing that unique minimum-weight perfect matching.

Let $w_{i,j}$ be the weight of edge (i,j) in G. Consider the skew-symmetric adjacency matrix $D = (d_{i,j})$ defined as follows: for $i < j$, $d_{i,j} = 2^{w_{i,j}}$ if (i,j) is an edge and $d_{i,j} = 0$ if (i,j) is not an edge; for $i > j$, $d_{i,j} = -d_{j,i}$. Then the following facts hold:

1. If there is a unique minimum-weight perfect matching M with weight W, then $\det(D) \neq 0$; moreover, the highest power of 2 dividing $\det(D)$ is 2^{2W}.
2. Furthermore, edge (i,j) is in M if and only if $\frac{\det(D_{i,j})2^{w_{i,j}}}{2^{2W}}$ is odd where $D_{i,j}$ is obtained by deleting rows i,j and columns i,j of D.

In our case the weights belong to the input, so the Isolating Lemma does not help. However, we can still use the above reconstruction procedure to construct some perfect matching which is potentially of minimal weight, and then test the uniqueness separately.

Constructing the unique minimum-weight perfect matching. For the first part of our algorithm (finding a symmetric permutation matrix associated to a perfect matching), unfortunately we cannot argue as elegantly as in the case of unweighted graphs that we treated in the last section, if we follow the reconstruction procedure by [13]. The reason is that we need values of a $\mathbf{GapL}$ function modulo 2^k for some k, and $\mathbf{GapL}$ is not known to be closed under integer division. Thus a $\mathbf{L}^{\mathbf{C_=L}}$ upper bound as in the unweighted case may not hold by this way. Instead, we describe another method for computing the unique minimum-weight perfect matching.

Let x be an indeterminate. We relabel all the edges (i, j) of G with $x^{w_{i,j}}$. Let $G(x)$ be the new graph and $A(x)$ its Tutte matrix. Then $\det(A(x))$ is a polynomial, $p(x) = \det(A(x)) = c_N x^N + c_{N-1}x^{N-1} + \cdots + c_0,\ c_N \neq 0$. Note that the degree N of p is bounded by the sum of all edge-weights of G. Thus when all the weights $w_{i,j}$ of G are polynomially bounded, N is also polynomially bounded, and all coefficients of $p(x)$ can be computed in **GapL**.

Assume for a moment that graph G has the unique minimum-weight perfect matching M with weight W. Observe that M corresponds to the lowest term x^{2W} in $p(x)$, moreover we have $c_{2W} = 1$ and $c_i = 0$, for all $0 \leq i < 2W$.

We denote by $G_{i,j}(x)$ the graph obtained from $G(x)$ by deleting the edge (i, j) and by $A_{i,j}(x)$ the Tutte matrix associated with $G_{i,j}(x)$. Furthermore, let $p_{i,j}(x) = \sum_{k\geq 0} c_k^{(i,j)} x^k = \det(A_{i,j}(x))$. Observe that

- If $(i, j) \in M$ then $c_{2W}^{(i,j)} = c_t^{(i,j)} = 0$, for all $0 \leq t \leq 2W$, because M can not be the unique minimum-weight perfect matching in $G - (i, j)$ which is obtained from G by deleting the edge (i, j). Graph $G - (i, j)$ has potentially other perfect matchings with weights bigger than W.
- If $(i, j) \notin M$ then $c_{2W}^{(i,j)} = 1 = c_{2W}$ and $c_t^{(i,j)} = 0$, for all $0 \leq t < 2W$, because M remains as the unique minimum-weight perfect matching in $G - (i, j)$.

Let $A = (a_{i,j})$ be the adjacency matrix of G. Define symmetric matrices $B_t = \left(b_{i,j}^{(t)}\right)$ by

$$b_{i,j}^{(t)} = b_{j,i}^{(t)} = a_{i,j} \sum_{k=0}^{t} \left(c_k - c_k^{(i,j)}\right), \text{ for } 0 \leq t \leq N. \tag{2}$$

It is clear that the elements of B_t are **GapL**-computable. As a consequence of the above observations we have the following fact.

Fact 3 *If G has a perfect matching M with minimal weight W, then B_{2W} is a symmetric permutation matrix and $B_t = \mathbf{0}$, for all $0 \leq t < 2W$.*

Since all elements of matrices B_t are computable in **GapL**, in $\mathbf{C_=L}$ we can test if B_t is a permutation matrix or a zero-matrix.

Lemma 8. *If G has unique minimum-weight perfect matching, then there exists $0 \leq t \leq N$ that $c_t = 1$, $c_s = 0$, B_t is a symmetric permutation matrix, and $B_s = \mathbf{0}$, for all $0 \leq s < t$. All these conditions can be tested in $\mathbf{C_=L}$.*

Testing uniqueness. The second part of our algorithm is to test if a given perfect matching M is unique with minimal weight in G. For weighted bipartite graphs we can develop an **NL**-algorithm that has B as input, where B is the permutation matrix associated to M. In analogy to the unweighted version of UPM, we don't know whether there is an **NC**-algorithm for weighted non-bipartite UPM.

In the bipartite case we look for an alternating cycle C (with respect to M) where the edges not in M have the same or less total weight than the edges of

M. If such a cycle exists, then $M \triangle C$ gives another matching M' with weight no more than that of M. If no such cycle exists, then M is the unique minimum-weight perfect matching in G. Algorithm ALT-WEIGHTED-CYCLE nondeterministically searches for such cycles.

With oracle access to B, the algorithm is in **NL**. Note that here too it is crucial that weights are given in unary; thus the cumulative weights a_M and $a_{\overline{M}}$ can be stored on a logspace tape.

Lemma 9. *Algorithm* ALT-WEIGHTED-CYCLE *tests correctly if a given perfect matching M is unique with minimal weight in a bipartite graph. It can be implemented in* **NL**.

```
ALT-WEIGHTED-CYCLE$(G, B)$

1  $a_M \leftarrow 0$; $a_{\overline{M}} \leftarrow 0$
2  guess $s \in V$
3  $i \leftarrow s$
4  repeat
5      guess $j \in V$
6      if $b_{i,j} = 0$ then reject
7      guess $k \in V \setminus \{i\}$
8      if $a_{j,k} = 0$ then reject
9      $a_M \leftarrow a_M + w_{i,j}$
10     $a_{\overline{M}} \leftarrow a_{\overline{M}} + w_{j,k}$
11     $i \leftarrow k$
12 until $k = s$
13 if $a_{\overline{M}} \le a_M$ then reject
14 accept
```

By combining Lemma 8 and 9 we can test if a bipartite graph G has unique minimum-weight perfect matching. Namely, the algorithm computes all matrices B_t, then it searches the potential perfect matching M by Lemma 8. Thereafter G and M are the inputs for algorithm ALT-WEIGHTED-CYCLE. By this way we can show that the weighted-bipartite UPM is in $\mathbf{NL}^{\mathbf{C_=L}}$ which is equal to $\mathbf{L}^{\mathbf{C_=L}}$.

In analogy to the unweighted case we can modify the computation of the perfect matching M by $B'_t = B_t \pmod 2$. The matrices B'_t are computed in $\oplus\mathbf{L}$. The rest of the algorithm is the same, giving an upper bound of $\mathbf{NL}^{\oplus\mathbf{L}}$.

Theorem 4. Weighted-bipartite UPM *with polynomially bounded weights is in* $\mathbf{L}^{\mathbf{C_=L}} \cap \mathbf{NL}^{\oplus\mathbf{L}}$.

5 Unique Perfect Matching is hard for NL

Chandra, Stockmeyer, and Vishkin [4] have shown that the perfect matching problem is hard for **NL**. We modify their reduction to show that UPM is hard for **NL**. Recall that UPM might be an easier problem than the general perfect matching problem.

Let $G = (V, E)$ be a directed acyclic graph, and let $s, t \in V$ be two vertices. By $\#\,\text{path}(G, s, t)$ we denote the number of paths in G from s to t. The connectivity problem asks whether $\#\,\text{path}(G, s, t) > 0$ and it is complete for **NL**. Since **NL** is closed under complement [8, 16], asking whether $\#\,\text{path}(G, s, t) = 0$ is also complete for **NL**.

In [4], the following undirected graph H is constructed: $H = (V_H, E_H)$,
$V_H = \{s_{out}, t_{in}\} \cup \{\, u_{in}, u_{out} \mid u \in V \setminus \{s,t\}\,\}$
$E_H = \{\,(u_{in}, u_{out}) \mid u \in V \setminus \{s,t\}\,\} \cup \{\,(u_{out}, v_{in}) \mid (u,v) \in E\,\}$
It is easy to see that $\#\,\mathrm{path}(G,s,t) = \#\,\mathrm{pm}(H)$. Therefore s_{out} is connected to t_{in} if and only if $H \in \mathrm{PM}$.

Now obtain H' from H by adding the edge (s_{out}, t_{in}). Observe that H' has at least one perfect matching, namely $M_H = \{(s_{out}, t_{in})\} \cup \{\,(u_{in}, u_{out}) \mid u \in V - \{s,t\}\,\}$. Other than this, H' and H have the same perfect matchings. We conclude that $\#\,\mathrm{pm}(H) + 1 = \#\,\mathrm{pm}(H')$. In summary, $\#\,\mathrm{path}(G,s,t) = 0 \iff \#\,\mathrm{pm}(H') = 1$. Note that each edge in H' is of the form (u_{in}, v_{out}); thus the partition $S, V_H \setminus S$ where $S = \{u_{in} \mid u_{in} \in V_H\}$ witnesses that H' is bipartite.

Theorem 5. UPM *is hard for* **NL***, even when restricted to bipartite graphs.*

As a consequence of the hardness of UPM, we consider the problem of testing if a given perfect matching M is unique in a graph G. The problem for bipartite graphs can be solved in **NL** by Lemma 4. For non-bipartite graphs we don't know whether the considered problem is in **NC** (note that if this problem for non-bipartite graph is in **NC**, then UPM for non-bipartite graphs is also in **NC**, because the unique perfect matching can always be computed in **NC**). Furthermore, the problem is hard for **NL** because in the above construction, $\#\,\mathrm{path}(G,s,t) = 0$ if and only if the perfect matching M_H is unique in the constructed graph H'. Thus we have

Corollary 2. *The problem of testing if a given perfect matching is unique in a bipartite graph is complete for* **NL***.*

Summary and Open Problems

We showed in the paper that the unique perfect matching problem for bipartite graphs for both cases weighted or unweighted is in **NC**. We have placed bipartite UPM between **NL** and $\mathbf{C_{=}L} \cap \mathbf{NL}^{\oplus \mathbf{L}}$ and the unique minimum-weight perfect matching problem between **NL** and $\mathbf{L}^{\mathbf{C_{=}L}} \cap \mathbf{NL}^{\oplus \mathbf{L}}$. Some questions remain open: 1) *Is non-bipartite* UPM *in* **NC***?* 2) *Can we improve the lower bound* **NL** *for* UPM*?* A possible improvement seems to be important because if UPM is hard for $\mathbf{C_{=}L}$, we could conclude that $\mathbf{C_{=}L} \subseteq \mathbf{NL}^{\oplus \mathbf{L}}$ (which is an open question), if UPM is hard for $\oplus\mathbf{L}$, we could conclude that $\oplus\mathbf{L} \subseteq \mathbf{L}^{\mathbf{C_{=}L}}$ (which is open too).

The same question about the upper bound can be asked for weighted non-bipartite graphs. Also, we restricted the weights to be polynomially bounded. It is not clear how to handle exponential weights. The current technique to determine one perfect matching would then lead to double exponential numbers. This is no longer in $\mathbf{NC}^2$. Note however, that the weight of an alternating cycle requires summing up at most n weights, which can be done in $\mathbf{NC}^2$.

Note that the results involving $\oplus\mathbf{L}$, namely Lemma 6, Corollary 1 and Theorem 4, carry over to $\mathbf{Mod_p L}$ for any p as well. The open questions listed above concerning $\oplus\mathbf{L}$ are open for all these classes as well.

Clearly, the most important open problem is: *Is the perfect matching problem in* **NC***?*

Acknowledgments

We thank Jochen Messner and Ilan Newman for very interesting discussions. Eric Allender gave very helpful comments on an earlier version of the paper. We also thank the anonymous referees for pointing out the generalization to $\mathbf{Mod_pL}$, and for comments which improved the presentation.

References

1. E. Allender, V. Arvind, and M. Mahajan. Arithmetic complexity, Kleene closure, and formal power series. *Theory Comput. Syst.*, 36(4):303–328, 2003.
2. E. Allender, R. Beals, and M. Ogihara. The complexity of matrix rank and feasible systems of linear equations. *Computational Complexity*, 8(2):99–126, 1999.
3. E. Allender, K. Reinhardt, and S. Zhou. Isolation, matching and counting: uniform and nonuniform upper bounds. *Journal of Computer and System Sciences*, 59:164–181, 1999.
4. A. Chandra, L. Stockmeyer, and U. Vishkin. Constant depth reducibility. *SIAM Journal on Computing*, 13(2):423–439, 1984.
5. J. Edmonds. Maximum matching and a polyhedron with 0-1 vertices. *Journal of Research National Bureau of Standards*, 69:125–130, 1965.
6. H. N. Gabow, H. Kaplan, and R. E. Tarjan. Unique maximum matching algorithms. In *31st Symposium on Theory of Computing (STOC)*, pages 70–78. ACM Press, 1999.
7. T. M. Hoang and T. Thierauf. The complexity of the characteristic and the minimal polynomial. *Theoretical Computer Science*, 295:205–222, 2003.
8. N. Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on Computing*, 17(5):935–938, 1988.
9. R. M. Karp, E. Upfal, and A. Wigderson. Constructing a perfect matching is in random NC. *Combinatorica*, 6:35–48, 1986.
10. D. Kozen, U. Vazirani, and V. Vazirani. NC algorithms for comparability graphs, interval graphs, and testing for unique perfect matching. In *Proceedings of FST&TCS Conference, LNCS Volume 206*, pages 496–503. Springer-Verlag, 1985.
11. D. Kozen, U. Vazirani, and V. Vazirani. NC algorithms for comparability graphs, interval graphs, and testing for unique perfect matching. Technical Report TR86-799, Cornell University, 1986.
12. L. Lovasz. On determinants, matchings and random algorithms. In L. Budach, editor, *Proceedings of Conference on Fundamentals of Computing Theory*, pages 565–574. Akademia-Verlag, 1979.
13. K. Mulmuley, U. Vazirani, and V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–131, 1987.
14. M. Rabin and V. Vazirani. Maximum matchings in general graphs through randomization. *Journal of Algorithms*, 10(4):557–567, 1989.
15. J. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27:701–717, 1980.
16. R. Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Informatica*, 26(3):279–284, 1988.
17. J. E. Tabaska, R. B. Cary, H. N. Gabow, , and G. D. Stormo. An RNA folding method capable of identifying pseudoknots and base triples. *Bioinformatics*, 14(8):691–699, 1998.
18. R. Zippel. Probabilistic algorithms for sparse polynomials. In *International Symposium on Symbolic and Algebraic Computation, LNCS 72*, pages 216–226, 1979.