On the unique perfect matching problem

Thanh Minh Hoang*
Abt. Theor. Inform.
Universität Ulm
89069 Ulm, Germany

Meena Mahajan
Inst. of Math. Sciences
Chennai 600 113, India

Thomas Thierauf
FB Elektr. und Inform.
FH Aalen
73430 Aalen, Germany

Abstract

Lovász has posed the question of whether the problem of testing if a graph has precisely one perfect matching is in **NC**. This *unique perfect matching* question has been answered positively by Kozen, Vazirani, and Vazirani for bipartite graphs.

In this note, we give tighter bounds on the complexity of the unique perfect matching problem. We show that the problem for bipartite graphs is in $\mathbf{C_=L}$ and in $\mathbf{NL}^{\oplus\mathbf{L}}$, a subclass of $\mathbf{NC}^2$. By a counterexample we show that the solution for the bipartite graphs can not be applied for non-bipartite graphs.

We also consider the weighted version of the problem. We show that the unique 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 the unique perfect matching problem is hard for **NL**, even when restricted to bipartite graphs.

1 Introduction

The perfect matching problem (short: PM) asks whether there exists a perfect matching in a given graph. This is one of the most prominent open questions in complexity theory regarding parallel computations. It is still open whether there is an **NC**-algorithm for deciding the existence of some perfect matchings in a given graph. PM is known to be in **P** by Edmonds [Edm65], and in randomized **NC** (short: **RNC**) by Lovász [Lov79]; subsequently Karp, Upfal and Wigderson [KUW86], and then Mulmuley, Vazirani, and Vazirani [MVV87] showed that constructing a perfect matching, if one exists, is in $\mathbf{RNC}^3$ and $\mathbf{RNC}^2$, respectively. Recently, Allender, Reinhardt, and Zhou [ARZ99] showed that PM is in non-uniform **SPL**.

Since no **NC**-algorithm for PM is known today, some special cases of the problem attract great attention. For example, an **NC** upper bound has been found for regular bipartite graphs [LPV81], dense graphs [DHK93], strongly chordal graphs [DK86] and planar graphs [Kas67, Vaz89]. For graphs with a polynomially bounded number of perfect matchings, PM is also in **NC** [GK87].

The problem of constructing some perfect matchings in a graph has been studied intensively. The construction version of PM is also in **P**, but no **NC** algorithm for it is known today. Mulmuley, Vazirani and Vazirani [MVV87]. showed that the problem is in $\mathbf{RNC}^2$.

In this paper we consider the complexity of the unique perfect matching problem by Lovász (see [KVV85]), 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 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 [TCGS98]).

Gabow, Kaplan, and Tarjan [GKT99] showed that UPM is in **P**. Kozen, Vazirani, and Vazirani [KVV85, KVV86] showed that UPM for bipartite graphs is in **NC**. Their techniques don't seem to generalize to

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

arbitrary graphs (see Section 3.2 for more detail). [1]

In this paper we give tighter bounds on the complexity of bipartite UPM. Our bounds place bipartite UPM into small logspace counting complexity classes. In Section 3 we show bipartite UPM is in $\mathbf{C_{=}L} \cap \mathbf{NL}^{\oplus\mathbf{L}}$. We show in Section 4 that 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 UPM is easier than the perfect macthing problem. However, we show in Section 5 that UPM is hard for **NL**, and it remains hard even when restricted to bipartite graphs. Thus the best known lower bounds for PM and for UPM coincide. Our results thus place bipartite UPM between **NL** and $\mathbf{C_{=}L}$. Furthermore, our results provides a new complete problem for **NL**. This is the problem of testing if a given perfect matching is unique in a bipartite graph.

2 Preliminaries

Complexity Classes Standard complexity classes are languages accepted by detereministic or nondeterministic logspace bounded Turing machines, denoted by **L** and **NL**, respectively. We give the definition of some counting classes. 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 **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 for all x we have $x \in A \iff f(x) = 0$. $\mathbf{C_{=}L}$ is closed under union and intersection, but it is not known to be closed under complement. The class $\oplus\mathbf{L}$ is the class of sets A for which there exists a function $f \in \mathbf{GapL}$ such that for all x we have $x \in A \iff f(x) \equiv 0 \pmod 2$.
parityL is closed under Turing reductions.

Some known relationships among these classes and their relativized versions are as follows:

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

Circuit classes $\mathbf{NC}^k$ are all families of languages or funtions that can be computed by polynomial-size circuits of depth $O(\log^k n)$.

The study of **GapL** and $\mathbf{C_{=}L}$ is motivated by the result that computing the determinant of an integer matrix is complete for **GapL** [Dam91, Tod91, Vin91, Val92]. As a consequence, testing the singularity of a matrix is complete for $\mathbf{C_{=}L}$, and computing the matrix rank is complete for $\mathbf{L}^{\mathbf{C_{=}L}}$ [ABO99].

Perfect Matchings in a Graph Let $G = (V, E)$ be an undirected graph with vertex set V and edge set E. A *matching* in G is a edge subset $M \subseteq E$ 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 all perfect matchings in G.

The perfect matching problem is defined by the set $\mathrm{PM} = \{\, G \mid \#\mathrm{pm}(G) > 0 \,\}$. The unique perfect matching problem is defined by $\mathrm{UPM} = \{\, G \mid \#\mathrm{pm}(G) = 1 \,\}$. Restricted to bipartite graphs, we call 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 following matrix $A = (a_{i,j})_{1 \le i,j \le n}$ is called *skew-symmetric adjacency matrix*:

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

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

[1] The conference version of [KVV85] claims that the technique works also for the general case. In a later version of the paper [KVV86] which is available on the home page of Dexter Kozen this is no longer claimed.

Theorem 2.1 (Tutte 1952) *Graph G has a perfect matching if and only if* $\det(A(x)) \neq \mathbf{0}$.

Since $\det(A(X))$ is a symbolic multivariate polynomial, it can have exponential length in n, when written as a sum of monoms. However, there are randomized identity tests for polynomials that just need to evaluate a polynomial at a random point [Zip79, Sch80]. Since the determinant of an integer matrix is complete for **GapL**, a subclass of $\mathbf{NC}^2$, Tuttes Theorem puts PM in $\mathbf{RNC}^2$.

It is well known from linear algebra that, for an $n \times n$ skew-symmetric matrix ($A = -A^T$), we have $det(A) = 0$ 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 2.2 *1.* $\#\mathrm{pm}(G) = 0 \implies \det(A) = 0$,

2. $G \in \mathrm{UPM} \implies \det(A) = 1$.

Rabin and Vazirani [RV89] used Fact 2.2 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:

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

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 $\mathbf{L}^{\mathbf{C_=L}}$

It is known from the last section that if $G \in \mathrm{UPM}$ then the unique perfect matching M in G can be easily computed. We represent M by the following matrix $B = (b_{i,j})$ of order n, whre

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

Note that $b_{i,j} \geq 0$ because $A_{i,j}$ is skew symmetric.

From the discussion above we have

Lemma 3.1 $G \in \mathrm{UPM} \implies$ *B is a symmetric permutation matrix.*

The first step of our algorithm for UPM is to check that, for a given graph G, matrix B is indeed a permutation matrix. We may assume that G is undirected. Therefore matrix B is symmetric. Hence B is a permutation matrix if and only if every row contains precisely one 1 and the rest is 0. This is equivalent to

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

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 3.2 $\{G \mid$ *B is a permutation matrix*$\} \in \mathbf{C_=L}$.

If matrix B is not a permutation matrix, we can already tell that $G \notin \mathrm{UPM}$. So assume that B is a permutation matrix, and therefore defines a perfect matching M in G. Suppose there is another perfect matching M' in G. It is well known that the graph $(V, M \triangle M')$ is a union of disjoint alternating cycles. That is, each cycle has even length and has alternating one edge in M and one edge in M'.

Definition 3.3 *Let M be a perfect matching in G. An* alternating cycle in G with respect to M *is a cycle of even length that has alternating one edge in M and one edge not in M.*

Lemma 3.4 *Let M be a perfect matching in G. $G \in$* UPM *if and only if there is* no *alternating cycle in G with respect to M.*

Alternating Cycle(G)

```
 1  guess s ∈ V
 2  i ← s
 3  repeat
 4      guess j ∈ V
 5      if b_{i,j} = 0 then reject
 6      guess k ∈ V
 7      if a_{j,k} = 0 or b_{j,k} = 1 then reject
 8      i ← k
 9  until k = s
10  accept
```

The nondeterministic algorithm Alternating Cycle searches for an alternating cycle in $G = (V, E)$ with respect to the perfect matching M defined by B. 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 of j which must be different from i such that $(j, k) \notin M$ (line 6 and 7). If $k = s$ in line 9 we closed an alternating cycle of length at least 4.

Alternating Cycle can be implemented on a nondeterministic logspace bounded Turing machine that has access to the following oracle set $S \in \mathbf{C_=L}$:

$$S = \{\, (G, i, j, c) \mid b_{i,j} = c \,\}.$$

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.

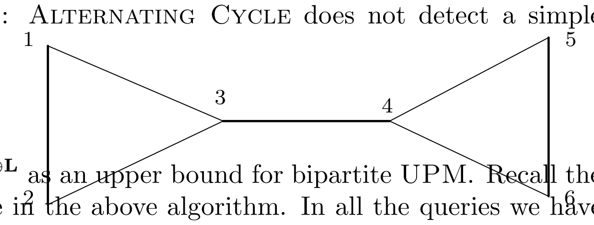
Note that $\mathbf{NL}^{\mathbf{C_=L}} = \mathbf{L}^{\mathbf{C_=L}}$ [ABO99]. Therefore we have

Lemma 3.5 *Let G be a graph such that B is a permutation matrix. Then in $\mathbf{L}^{\mathbf{C_=L}}$ we can decide whether G is in* UPM.

We combine the algorithms from Lemma 3.2 and 3.5 and obtain an algorithm to decide whether $G \in$ UPM for a given graph G that works in $\mathbf{L}^{\mathbf{C_=L}}$.

Theorem 3.6 *Bipartite* UPM $\in \mathbf{L}^{\mathbf{C_=L}}$.

Unfortunately, Alternating Cycle works correctly only for bipartite graphs. Since bipartite graphs do not have any odd cycles, each 'alternating cycle' found by Alternating Cycle is properly simple. But the situation for non-bipartite graphs is different: Alternating Cycle does not detect a simple cycle and therefore it produces possibly a false answer. The graph show to the right provides an example. The unique perfect matching is $M = \{(1, 2), (3, 4), (5, 6)\}$, but Alternating Cycle outputs 'accept'.



It is interesting to note that we can also obtain $\mathbf{NL}^{\oplus\mathbf{L}}$ as an upper bound for bipartite UPM. Recall the oracle set $S = \{\, (G, i, j, c) \mid |a_{i,j}| \det(A_{i,j}) = c \,\}$ we use in the above algorithm. In all the queries we have $c = 0$ or $c = 1$. Suppose we replace S by the following set $T \in \oplus\mathbf{L}$:

$$T = \{\, (G, i, j, c) \mid |a_{i,j}| \det(A_{i,j}) \equiv c \pmod 2 \,\}.$$

Algorithm Permutation checks that B is a permutation matrix over $\mathbf{Z}_2$. It can be implemented on a deterministic logspace bounded Turing machine that has access to the oracle set T. Note that $\mathbf{L}^{\oplus\mathbf{L}} = \oplus\mathbf{L}$.

```
PERMUTATION(G)
for i ← 1 to n do
    k ← 0
    for j ← 1 to n do
        if b_{i,j} ≡ 1 (mod 2) then
            if k = 0 then k ← 1
            else reject
accept
```

Lemma 3.7 *$\{G \mid B$ 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 3.8 *Bipartite* UPM $\in \mathbf{NL}^{\oplus\mathbf{L}}$.

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

Based on the method in [KVV86], 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. Then the skew-symmetric adjacency matrix of G is of the form $S = \begin{bmatrix} \mathbf{0} & A \\ -A^T & \mathbf{0} \end{bmatrix}$.

Since $\det(S) = \det^2(A)$, by Fact 2.2 we obtain the following fact for computing the unique perfect matching in the bipartite graph G:

Fact 3.9 *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 [KVV85] in such a way that we get $\mathbf{C_=L}$ as an upper bound.

Lemma 3.10 *For bipartite graph G with $2n$ vertices and bipartite adjacency matrix A, define matrices $B = [b_{i,j}]$ and C 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 identity matrix and $A_{i|j}$ is the sub-matrix obtained by deleting the i-th row and the j-th column of A. The bipartite graph G has a unique perfect matching if and only if

(i) B is a permutation matrix, and

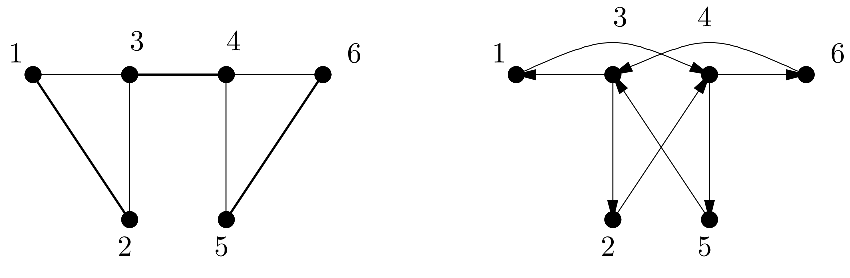
(ii) the characteristic polynomial of C is $\chi_C(x) = x^n$.

We provide some intuition to the lemma. Just as in the general case of Lemma 3.2, 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. 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.

We consider the complexity of checking the conditions of Lemma 3.10. 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 3.2. Consider the condition (ii). Here the elements of matrix C are not given as input, they are certain determinants. However, a result in [AAM03] 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 $\mathbf{C_=L}$. Therefore condition (ii) can be checked in $\mathbf{C_=L}$. We conclude:

Theorem 3.11 *Bipartite* UPM *is in* $\mathbf{C_{=}L}$.

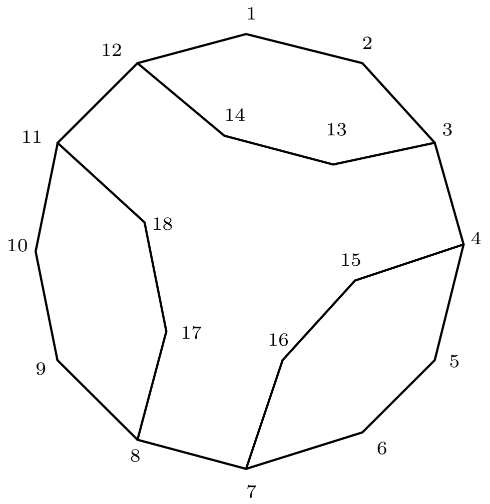
Unfortunately, the above technique doesn't seem to generalize to non-bipartite graphs. (The conference version of [KVV85] claims that the technique works also for the general case.) The graph G shown in the figure provides an example where the technique doesn't seem to work. Observe that $G \in$ UPM 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

Consider the concept of the skew-symmetric matrix of a non-bipartite graph, we ask whether there is an analog to Lemma 3.10. Given a perfect matching M in an undirected graph G having n vertices and the skew-symmetric adjacency matrix A. Let $B = (b_{i,j})$ be the skew-symmetric matrix associated to M, i.e. B is obtained from the symmetric permutation matrix corresponding to M by changing the signs of the elements under the main diagonal. Define $C = I - AB^T$.

Now the analog claim to Lemma 3.10 would be that the perfect matching M is unique in G if and only if $\chi_C(x) = x^n$. Unfortunately, this is not true. The graph G shown to the right provides a counterexample. Obviously, G is not in UPM. The following properties have been checked by Maple: the determinant of the skew-symmetric matrix A is equal to 1, its minors are as follows: $\det(A_{1,2|1,2}) = \det(A_{3,4|3,4}) = \det(A_{5,6|5,6}) = \det(A_{7,8|7,8}) = \det(A_{9,10|9,10}) = \det(A_{11,12|11,12}) = \det(A_{13,14|13,14}) = \det(A_{15,16|15,16}) = \det(A_{17,18|17,18}) = 1$ and $\det(A_{1,12|1,12}) = \det(A_{2,3|2,3}) = \det(A_{4,5|4,5}) = \det(A_{6,7|6,7}) = \det(A_{8,9|8,9}) = \det(A_{10,11|10,11}) = \det(A_{12,14|12,14}) = \det(A_{13,3|13,3}) = \det(A_{4,15|4,15}) = \det(A_{7,16|7,16}) = \det(A_{8,17|8,17}) = \det(A_{8,11|8,11}) = 0$. Therefore the matrix computed by the above minors is the permutation matrix B which corresponds to the perfect matching $M = \{(1,2),(3,4),(5,6),(7,8),(9,10),(11,12),(13,14),(15,16),(17,18)\}$. For $C = I - AB^T$ we get $\chi_C(x) = x^{18}$.



4 Unique minimum-weight perfect matching

We consider graphs with positive edge-weights in this section. 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 [MVV87] that there is a $\mathbf{RNC}^2$ algorithm (by using Isolating Lemma) for computing some perfect matching in graph G. This randomized algorithm chooses in random weights for the edges of G, then the Isolating Lemma states that there is a unique perfect matching of minimal weight with high probability. We describe briefly the procedure by [MVV87] 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 [MVV87]. The reason is that we need values of a **GapL** function modulo 2^k for some k, and **GapL** is not known to be closed under integer division. Thus a $\mathbf{L}^{\mathbf{C}_=\mathbf{L}}$ upper bound as in the unweighted case as 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 $x^{w_{i,j}}$, i.e.

$$w_{i,j} \longmapsto x^{w_{i,j}}.$$

Let $G(x)$ be the new graph and $A(x)$ its Tutte matrix. Then $\det(A(x))$ is a polynomial, $p(x) = \det(A(x)) = c_N x^N + c_{N-1}x^{N-1} + \cdots + c_0,\ c_N \neq 0$. In the case when all the weights $w_{i,j}$ of G are polynomially bounded, we have that the coefficients c_i of p are polynomially bounded. The same holds for the degree N of p, which is bounded by the sum of all edge-weights of G.

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 $0 \leq i \leq 2W$.

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 biger than W.

- If $(i,j) \notin M$ then $c_{2W}^{(i,j)} = 1 = c_{2W}$ and $c_t^{(i,j)} = 0$, for all $0 \leq t \leq 2W$, because M remains to be 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

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

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 4.1 *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 $0 \leq t \leq 2W$.*

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

Lemma 4.2 *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 $\mathbf{C}_=\mathbf{L}$.*

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 ALTERNATING-WEIGHTED-CYCLE nondeterministically searches for such cycles.

With oracle access to B, the procedure is in **NL**. Note that here too it is crucial that weights are given in unary; thus the cumulative weights m_1 and m_2 can be stored on a logspace tape.

$m_1 \leftarrow 0$; $m_2 \leftarrow 0$
guess $s \in V$
$i \leftarrow s$
repeat
    guess $j \in V$
    if $b_{i,j} = 0$ **then reject**
    $m_1 \leftarrow m_1 + w_{i,j}$
    guess $k \in V$
    if $a_{j,k} = 0$ **or** $b_{j,k} = 1$ **then reject**
    $m_2 \leftarrow m_2 + w_{j,k}$
    $i \leftarrow k$
until $k = s$
if $m_2 \leq m_1$ **then reject**
accept

ALTERNATING-WEIGHTED-CYCLE(G, B)

Lemma 4.3 *Algorithm* ALTERNATING-WEIGHTED-CYCLE *tests correctly if a given perfect matching M is unique with minimal weight in a bipartite graph. It can be done in* **NL**.

By combining Lemma 4.2 and 4.3 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 4.2. Thereafter G and M are the inputs for Procedure AWC. 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 remains unchanged. Thus we can get $\mathbf{NL}^{\oplus\mathbf{L}}$ as an upper bound for weighted-bipartite-UPM.

Theorem 4.4 Weighted-bipartiteUPM $\in \mathbf{L}^{\mathbf{C_=L}} \cap \mathbf{NL}^{\oplus\mathbf{L}}$ *(polynomially bounded weights).*

5 Unique Perfect Matching is hard for NL

Chandra, Stockmeyer, and Vishkin [CSV84] 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 a graph G going from s to t. The connectivity problem asks whether $\#\text{path}(G, s, t) > 0$. It is complete for **NL**, as well as its complement that asks whether $\#\text{path}(G, s, t) = 0$.

Chandra, Stockmeyer, and Vishkin [CSV84] constructed an undirected graph H as follows:

The vertices of H are s_{out} and t_{in}, and for each vertex $v \in V - \{s, t\}$ graph H has two vetices v_{in} and v_{out}. The edges of H are as follows:

1. For each vertex $v \in V(G)$, there is an edge $v_{in}v_{out}$ in H.
2. For each edge $uv \in E(G)$, there is an edge $u_{out}v_{in}$ in H.

Then it is not hard to see that $\#\text{path}(G, s, t) = \#\text{pm}(H)$, where $\#\text{pm}(H)$ is the number of perfect matchings of H. Therefore s_{out} is connected to t_{in} if and only if $H \in \text{PM}$.

Now we modify H by adding the edge $s_{out}t_{in}$. Call the resulting graph H'. Observe that H' has at least one perfect matching: take the edge $s_{out}t_{in}$ and all the edges $u_{in}u_{out}$. Other than this, H' and H have the same perfect matchings. We conclude that $\#\mathrm{pm}(H) + 1 = \#\mathrm{pm}(H')$. In summary,

$$\#\mathrm{path}(G, s, t) = 0 \Longleftrightarrow \#\mathrm{pm}(H') = 1.$$

Note that H is acyclic, and all cycles in H' have even length. Therefore H' is bipartite.

Theorem 5.1 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 3.5. 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 be computed always in **NC**). Furthermore, the problem is hard for **NL** because we have in above that $\#\mathrm{path}(G, s, t) = 0$ if and only if the perfect matching $\{s_{out}t_{in}\} \cup \{u_{in}u_{out} \mid u \in V - \{s, t\}\}$ is unique in the constructed graph H'. Thus we have

Corollary 5.2 *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 **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 be improved 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 the weighted non-bipartite graphs. Also, we restricted the weights to be polynomially bounded. Moreover, 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 $\mathbf{NC}^2$.

Clearly, the precise complexity of the general perfect matching problem is the most important open problem, i.e. the question is: *Is the perfect matching problem in* **NC***?*

Acknowledgments

We want to thank Jochen Messner and Ilan Newman for very interesting discussions. Eric Allender gave very helpful comments on an earlier version of the paper.

References

[AAM03] E. Allender, V Arvind, and M. Mahajan. Arithmetic complexity, Kleene closure, and formal power series. *Theory of Computing Systems*, 36(4):303–328, 2003.

[ABO99] E. Allender, R. Beals, and M. Ogihara. The complexity of matrix rank and feasible systems of linear equations. *Computational Complexity*, 8:99–126, 1999.

[AO96] E. Allender and M. Ogihara. Relationship among **PL**, **#L**, and the determinant. *RAIRO-Theoretical Informatics and Applications*, 30:1–21, 1996.

[ARZ99] E. Allender, K. Reinhardt, and S. Zhou. Isolating, matching, and counting: uniform and nonuniform upper bounds. *Journal of Computer and System Sciences*, 59:164–181, 1999.

[CSV84] A. K. Chandra, L. Stockmeyer, and U. Vishkin. Constant depth reducibility. *SIAM Journal on Computing*, 13(2):423–439, 1984.

[Dam91] C. Damm. DET = $\mathrm{L}^{(\#\mathrm{L})}$. Technical Report Informatik-Preprint 8, Fachbereich Informatik der Humboldt Universitaet zu Berlin, 1991.

[DHK93] E. Dahlhaus, P. Hajnal, and M. Karpinski. On the parallel complexity of hamiltonian cycles and matching problem in dense graphs. *Journal of Algorithms*, 15:367–384, 1993.

[DK86] E. Dahlhaus and M. Karpinski. The matching problem for strongly chordal graphs is in **NC**. Technical Report 855-CS, University of Bonn, 1986.

[Edm65] J. Edmonds. Maximum matching and a polyhedron with 0-1 vertices. *Journal of Research National Bureau of Standards*, 69:125–130, 1965.

[GK87] D. Grigoriev and M. Karpinski. The matching problem for bipartite graphs with polynomially bounded permanent is in **NC**. In *28th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 166–172. IEEE Computer Society Press, 1987.

[GKT99] 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.

[Kas67] P. W. Kastelyn. Graph theory and crystal physics. In F. Harary, editor, *Graph Theory and Theoretical Physics*, pages 43–110. Academic Press, 1967.

[KUW86] R. M. Karp, E. Upfal, and A. Wigderson. Constructing a perfect matching is random **NC**. *Combinatorica*, 6(1):35–48, 1986.

[KVV85] D. C. Kozen, U. V. Vazirani, and V. V. Vazirani. **NC** algorithms for comparability graphs, interval graphs, and testing for unique perfect matchings. In *5th Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 496–503. Springer-Verlag, 1985.

[KVV86] D. C. Kozen, U. V. Vazirani, and V. V. Vazirani. **NC** algorithms for comparability graphs, interval graphs, and testing for unique perfect matchings. Technical Report TR86-799, Cornell University, 1986.

[Lov79] L. Lovász. On determinants, matchings and random algorithms. In *Conference on Fundamentals of Computing Theory*, pages 565–574. Akademia-Verlag, 1979.

[LPV81] G. Lev, M. Pippenger, and L. Valiant. A fast parallel algorithm for routing in permutation networks. *IEEE Transactions on Computers*, C-30:93–100, 1981.

[MVV87] K. Mulmuley, U. V. Vazirani, and V. V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7:105–113, 1987.

[RV89] M. O. Rabin and V. V. Vazirani. Maximum matching in general graphs through randomization. *Journal Algorithms*, 10:557–567, 1989.

[Sch80] J. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27:701–717, 1980.

[TCGS98] 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.

[Tod91] S. Toda. Counting problems computationally equivalent to the determinant. Technical Report CSIM 91-07, Dept. of Computer Science and Information Mathematics, University of Electro-Communications, Chofu-shi, Tokyo 182, Japan, 1991.

[Val79] L. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

[Val92] L. Valiant. Why is Boolean complexity theory difficult. In M. S. Paterson, editor, *Boolean Function Complexity*, London Mathematical Society Lecture Notes Series 169. Cambridge University Press, 1992.

[Vaz89] V. Vazirani. NC algorithms for computing the number of perfect matchings in $K_{3,3}$-free graphs and related problems. *Information and computation*, 80(2):152–164, 1989.

[Vin91] V Vinay. Counting auxiliary pushdown automata and semi-unbounded arithmetic circuits. In *6th IEEE Conference on Structure in Complexity Theory (CCC)*, pages 270–284. IEEE Computer Society Press, 1991.

[Zip79] R. Zippel. Probabilistic algorithms for sparse polynomials. In *International Symposium on Symbolic and Algebraic Computation*, Lecture Notes in Computer Science 72, pages 216–226. Springer-Verlag, 1979.