Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 201 (2010)

Perfect Matching in Bipartite Planar Graphs is in UL

Samir Datta * Raghav Kulkarni † Raghunath Tewari‡

January 4, 2011

Abstract

We prove that Perfect Matching in bipartite planar graphs is in UL, improving upon the previous bound of SPL (see [DKR10]) on its space complexity. We also exhibit space complexity bounds for some related problems. Summarizing, we show that, constructing:

1. a Perfect Matching in bipartite planar graphs is in UL
2. a Hall Obstacle in bipartite planar graphs is in NL;
3. an Even Perfect Matching in bipartite planar graphs is in NL; and
4. an Even Path in planar DAGs is in UL.

For the proof of 2 and 3, we revisit the flow technique of Miller and Naor [MN89] which was used to provide an NC algorithm for bipartite planar matching construction. To obtain the main result (item 1 above), we combine it in a simple but subtle way with the double counting technique of Reinhardt and Allender [RA00] to yield the UL bound.

To prove the UL bound in 4 above, we combine two existing isolation techniques (viz. [BTV09] and [FKS84, Hoa10]) in a non-obvious way.

1 Introduction

1.1 Matching Problems in Graphs

A *matching* M in a graph G is a set of vertex disjoint edges. The end-points of the edges in M are said to be *matched*. A *perfect matching* in a graph G is a matching M such that every vertex of G is matched. See [LP86] for an excellent introduction to matching and related problems. We consider the following computational problems related to matching:

- Perfect-Matching (Decision) : decide if G contains a perfect matching.
- Perfect-Matching (Construction) : construct a perfect matching in G (if exists).
- Min-Wt-PM (Decision) : given G together with edge-weights $w : E(G) \to \mathbb{Z}$ such that $|w(e)| \leq n^{O(1)}$, and an integer k - decide if G contains a perfect matching of weight at most k.

*Chennai Mathematical Institute, India: email:sdatta@cmi.ac.in

†University of Chicago. Part of the work was done while visiting Chennai Mathematical Institute and University of Nebraska-Lincoln: email:raghav@cs.uchicago.edu

‡University of Nebraska-Lincoln: email:rtewari@cse.unl.edu. Research supported in part by NSF grants CCF-0830730 and CCF-0916525.

ISSN 1433-8092

- MAX-MATCHING (DECISION) : given G and an integer k, decide whether or not G has a matching of cardinality at least k.
- UPM (DECISION) : decide whether or not G has a unique perfect matching

The Hall's Theorem (see for instance [LP86]) asserts that a bipartite graph $G = (A \cup B, E)$ has a perfect matching iff $|A| = |B|$ and for every $S \subseteq A : |N(S)| \geq |S|$, where $N(S) := \{v \in B \mid \exists u \in A : (u, v) \in E\}$. A *Hall-obstacle* in a bipartite graph $G = (A \cup B, E)$ is a set $S \subseteq A$ such that $|N(S)| < |S|$. We consider the following computational problems related to Hall-obstacle:

- HALL-OBS (DECISION) : decide if a bipartite G contains a Hall-obstacle.
- HALL-OBS (CONSTRUCTION) : construct a Hall-obstacle in a bipartite G (if exists).

Historically, matching problems have been occurring as central problems in Algorithms and Complexity Theory. Edmond's *blossom* algorithm [Edm65] for MAX-MATCHING was one of the first examples of a non-trivial polynomial time algorithm. It had a considerable share in initiating the study of *efficient computation*, including the class P itself; Valiant's #P-hardness [Val79] for counting perfect matchings in bipartite graphs provided surprising insights into the counting complexity classes. The study of whether or not PERFECT-MATCHING is well parallelizable has yielded powerful tools such as *isolating lemma* [MVV87] that have found numerous applications elsewhere.

The rich combinatorial structure of matching problems combined with their potential to serve as central problems in the field invites their study from several perspectives. The focus of this paper is on the space complexity of the matching problems. The best known upper bound for PERFECT-MATCHING (and other matching problems mentioned above) is *non-uniform* SPL [ARZ99] whereas the best hardness known is NL-hardness [CSV84].

1.2 Matching Problems in Planar Graphs

A well known example where planarity is a boon is that of counting perfect matchings. The problem in planar graphs is in P [Kas67] as opposed to being #P-hard in general graphs [Val79]. Counting perfect matchings in planar graphs can in fact be done in NC [Vaz88]; thus PERFECT-MATCHING (DECISION) in planar graphs is in NC. "Is PERFECT-MATCHING (CONSTRUCTION) in planar graphs in NC?" remains an outstanding open question, whereas the bipartite planar case is known to be in NC [MN89], [MV00], [KMV08], [DKR10].

The space complexity of matching problems in planar graphs was first studied by Datta, Kulkarni, and Roy [DKR10] where it was shown that MIN-WT-PM in bipartite planar graphs is in SPL. This result was recently generalized to bounded genus bipartite graphs by Datta, Kulkarni, Tewari, and Vinodchandran [DKTV10]. Kulkarni [Kul09] shows that MIN-WT-PM in planar graphs (not necessarily bipartite) is NL-hard. The only known hardness for PERFECT-MATCHING in planar graphs is L-hardness (see for instance [DKLM10]). For bipartite planar graphs, nothing better than L-hardness is known for any matching problem.

Given a directed graph G and two vertices s and t in G, let DIR-REACH denote the problem of deciding if there exists a path from s to t in G. DIR-REACH is NL-complete. It turns out that DIR-REACH in planar graphs reduces (in log-space) to PERFECT-MATCHING in bipartite planar graphs [DKLM10]; the former was proved to be in $\mathsf{UL} \cap \mathsf{coUL}$ by Bourke, Tewari, and Vinodchandran [BTV09]. In this paper we show that the later is in UL, leaving the coUL

Table 1: Space Complexity of Matching Problems in Planar Graphs

Problem in Planar Graphs	Upper Bound	Hardness
Perfect-Matching (Construction)	PSPACE [1]	L
Max-Matching (Decision)	PSPACE	L
Min-Wt-PM (Decision)	$\mathsf{L}^{\mathsf{C}_=\mathsf{L}}$ [Kas67]	NL [Kul09]
Perfect-Matching (Decision)	$\mathsf{L}^{\mathsf{C}_=\mathsf{L}}$ [Kas67]	L
bipartite Max-Matching	$\mathsf{L}^{\mathsf{C}_=\mathsf{L}}$ [Hoa10]	L
bipartite Min-Wt-PM	SPL [DKR10]	L
bipartite Hall-Obs (Construction)	NL (new)	L
bipartite Hall-Obs (Decision)	coUL (new)	L
bipartite Perfect-Matching	UL (new)	L
bipartite UPM	UL (new)	L [DKLM10]

bound as an intriguing open question. Table 1.2 records the current knowledge (including the results in this paper) about the space complexity of matching problems in planar graphs.

1.3 Our Results

Theorem 1.1. *In bipartite planar graphs:*
(a) Perfect-Matching *(*Decision + Construction*) is in* UL;
(b) Hall-Obs *(*Decision*) is in* coUL*;*
(c) Hall-Obs *(*Construction*) is in* NL.

We build on two key algorithms: (1) Miller and Naor's algorithm [MN89] for perfect matching in bipartite planar graphs; (2) Rienhardt and Allender's [RA00] UL algorithm for shortest path in *min-unique* graphs: graphs with polynomially bounded edge-weights and having at most one minimum weight path between any pair of vertices. Miller and Naor reduce the Perfect-Matching (Decision) in planar graphs to the following problem in directed planar graphs: Neg-Cycle (Decision) problem - given a directed graph with polynomially bounded edge-weights, decide whether or not the graph contains a negative weight cycle. We observe that this reduction works in log-space and Neg-Cycle problem is in NL. This yields the NL bound for perfect matching in bipartite planar graphs. For the proof of the UL bound in part (a), we first provide a technical extension of (2) when the graph contains negative weight edges but no negative weight cycles. A simple but subtle combination of (1) and (2) then yields the desired result. For part (b) and part (c), we argue that via a simple adaptation of Miller and Naor's algorithm, the Neg-Cycle directly corresponds to a Hall-obstacle. To the best of our knowledge this is the first time a bound on the space complexity (and the parallel complexity) of constructing Hall-obstacle in bipartite planar graphs is being noted. As opposed to [KMV08] and [DKR10], our space bounded algorithms do not require determinant computation as a subroutine, instead we make use of a variant of planar reachability. However, for the weighted case we do not know how to improve upon the SPL bound in [DKR10]. We also do not know how to improve on $\mathsf{L}^{\mathsf{C}=L}$ bound for maximum matching due to [Hoa10].

Let EXACT-PM (DECISION) denote the problem of deciding, given an integer k, whether or not a graph G with edges colored Red or Blue contains a perfect matching with exactly k Red edges. This problem was first posed by Papadimitriou and Yannakakis [PY82]. It is known to be in RNC [MVV87] but not known to be in P. We consider the following relaxation of the EXACT-PM problem: Let EVEN-PM (DECISION) denote the problem of deciding whether or not a graph G with edges colored Red or Blue contains a perfect matching with even number of Red edges. In this paper, we observe the following:

Theorem 1.2. *(a)* EVEN-PM *in bipartite graphs is in* P;
(b) EVEN-PM *in bipartite planar graphs is in* NL.

We also consider EVEN-PATH problem: deciding whether or not there is a directed path of even length between two specified vertices. EVEN-PATH is NP-complete [LP83] but restricted to planar graphs it is in P [Ned99]. In DAG, the problem is NL-complete. EVEN-PATH problem in planar DAG can be viewed as a relaxation of the following problem, which is NL-complete in planar DAG [Kul09]: RED-BLUE-PATH problem - given a directed graph with edges colored Red or Blue, decide whether or not there is a (simple) path between two specified vertices such that any two consecutive edges in the path are of two different colors.

Theorem 1.3. EVEN-PATH *in planar DAG is in* UL.

A motivation to study whether EVEN-PATH in planar DAG is in UL is the hope to develop new techniques to prove RED-BLUE-PATH in planar DAG is in UL, and thus NL = UL. Indeed, our proof of the UL bound for EVEN-PATH in planar DAG combines two different *isolation* techniques ([BTV09], [Hoa10]) in a non-obvious way. In the process, we also give a simple log-space procedure to obtain generalized BTV weights (see Weighting Scheme A in Section 5.2) in planar graphs without going through any piecewise linear embedding of the graph. Our procedure is inspired by the subroutine of computing *pseudo-flow* in Miller and Naor's algorithm and it might be of independent interest.

2 Preliminaries

2.1 Space Complexity Classes

Definition 1 (Complexity Classes)**.** - *The class* L *consists of the decision problems that can be solved using a deterministic* log-space *Turing machine.*

- *The class* NL *consists of the decision problems that can be solved using a non-deterministic* log-space *Turing machine.*

- *The class* UL *consists of the decision problems that are solvable by an* NL *machine with at most one accepting path.*

- *The class* $\#L$ *consists of functions of the form* $\#acc_M(x) : \Sigma^* \to N$ *(counting the number of accepting computations of an* NL *machine* M *on input* x*).*

- *The class* GapL *consists of functions that are the difference of two* #L *functions.*

- *The class* SPL *consists of those decision problems* $A \subseteq \Sigma^*$ *for which the characteristic vector* $\chi_A \in$ GapL, *where* $\chi_A \in \{0,1\}^{\Sigma^*}$ *indicates membership in* A.

- *The class $\oplus L$ consists of the decision problems $A \subseteq \Sigma^*$ defined as follows:*

$$\oplus L = \{A \mid \exists f \in \mathsf{GapL} \text{ such that: } x \in A \iff f(x) \equiv 1 \pmod 2.\}$$

See for instance [Vol99] for definitions of standard complexity classes. It is known that $\mathsf{UL} \subseteq \mathsf{NL} \subseteq \mathsf{NC} \subseteq \mathsf{P}$ and $\mathsf{UL} \subseteq \mathsf{SPL}$. It is also known that $\mathsf{SPL} \subseteq \oplus\mathsf{L} \subseteq \mathsf{NC}$. As of now, NL and SPL as well as NL and $\oplus\mathsf{L}$ are incomparable.

2.2 Flow Terminology

Here we rephrase the terminology used in [MN89].

An undirected *edge* is a two element unordered set $\{u, v\}$ such that $u, v \in V$. An undirected graph $G = (V, E)$ consists of a set V of *vertices* and a set E of *undirected edges*. An *arc* is an ordered tuple $(u, v) \in V \times V$. A directed graph $\overrightarrow{G} = (V, \overrightarrow{E})$ consists of a set V of *vertices* and a set $\overrightarrow{E} \subseteq V \times V$ of arcs. Given an undirected graph $G = (U, V)$, its directed version is a directed graph $\overleftrightarrow{G} = (V, \overleftrightarrow{E})$ where $\overleftrightarrow{E} := \{(u, v) \mid \{u, v\} \in E\}$.

A *capacity-demand graph* is a triple (G, c, d) where $G = (V, E)$; every arc $(u, v) \in \overleftrightarrow{E}$ is assigned a real value $c(u, v)$ called the *capacity* of the arc and every vertex $v \in V$ is assigned a real value $d(v)$ called the *demand* at the vertex.

A *pseudo-flow* in a capacity-demand graph (G, c, d) is a function $f : \overleftrightarrow{E} \to \mathbb{R}$ such that:
(i) for every arc $(u, v) \in \overleftrightarrow{E}$, we have:

$$\text{(skew-symmetry) } f(u, v) = -f(v, u),$$

and
(ii) for every vertex $v \in V$, we have:

$$\text{(demands met)} \sum_{w \in V : (v,w) \in \overleftrightarrow{E}} f(v, w) = d(v).$$

A *flow* in a capacity-demand graph (G, c, d) is a function $f : \overleftrightarrow{E} \to \mathbb{R}$ such that:
(a) f is a pseudo-flow in (G, c, d);
(b) for every $(u, v) \in \overleftrightarrow{E}$, we have:

$$\text{(capacity constraints satisfied) } f(u, v) \leq c(u, v).$$

A zero-demand graph (G, c) is a capacity-demand graph in which the demand at every vertex is zero.

2.3 Main Lemmas from Miller and Naor [MN89]

Definition 2 (Directed Dual)**.** *Let G be a planar graph. Fix an embedding of G in the plane. Let G^* denote the dual of G with respect to the fixed embedding. The directed dual of G is the directed version of G^* denoted by $\overleftrightarrow{(G^*)}$. The arcs of $\overleftrightarrow{G}$ and that of $\overleftrightarrow{(G^*)}$ are in one to one correspondence.*

Proposition 2.1 (folklore, see for instance [MN89])**.** *Let (G, c) be a zero-demand graph. Let f be a flow in (G, c). If $C^* = (e_1^*, \ldots, e_k^*)$ is a directed cycle in $\overleftrightarrow{(G^*)}$, then*

$$\sum_{e \,:\, e^* \in C^*} f(e) = 0.$$

Lemma 2.2 (Miller, Naor [MN89]). *If (G, c) be a zero-demand graph then: there exists a flow in (G, c) $\iff$ $\overleftrightarrow{(G^*)}$ has no negative weight cycle with respect to weights c.*

3 The NL Bounds

3.1 Decision Version

In this section, we describe the Miller and Naor's algorithm (Algorithm (I) in [MN89], Algorithm 2 below) for solving the decision version of the Perfect-Matching problem in bipartite planar graphs. We refer the reader to [MN89] for the proof of correctness of the algorithm. Our main observation is that the algorithm can be implemented in NL.

Constructing a pseudo-flow

Algorithm 1 gives a Log-space procedure to construct a pseudo-flow in a zero-demand graph. Miller and Naor (Algorithm (I) in [MN89]) use this as a subroutine to reduce Perfect-Matching (Decision) to Neg-Cycle (Decision) We observe that Neg-Cycle (Decision+ Construction) is in NL. This immediately yields an NL algorithm for Perfect-Matching (Decision).

Input : A capacity-demand graph (G, c, d)
Promise: $\sum_v d(v) = 0$
Output : A pseudo-flow in (G, c, d)

1 Compute a spanning tree T in G;
2 For any arc $(u, v) \notin \overleftrightarrow{T}$, set $f'(u, v) = 0$;
3 For an arc $(u, v) \in \overleftrightarrow{T}$, removing the edge $\{u, v\}$ separates the tree T into two subtrees. Let T_u denote the subtree containing u and T_v denote the subtree containing v. For any $(u, v) \in \overleftrightarrow{T}$, set $f'(u, v) = \sum_{w \in T_u} d(w)$;

Algorithm 1: MN-Pseudo-Flow [MN89]

Observation 3.1. *Constructing a pseudo-flow is in Log-space.*

Input : A bipartite planar graph $G = (A \cup B, E)$
Output : Yes if G has a perfect matching; No otherwise

1 Construct a capacity-demand graph (G, c, d) as follows: for each vertex $v \in A$, set $d(v) = 1$ and for each vertex $v \in B$, set $d(v) = -1$. For $u \in A, v \in B$, set $c(u, v) = 1$ and $c(v, u) = 0$;
2 Construct a pseudo-flow f' in (G, c, d);
3 Construct a zero-demand graph $(G, c - f')$;
4 Output Yes if $\overleftrightarrow{(G^*)}$ has no negative weight cycle with respect to weights $(c - f')$; Output No otherwise;

Algorithm 2: MN-Decision [MN89]

Observation 3.2. *Given a directed graph $\overrightarrow{G}$ with polynomially bounded weights on its arcs, the problem of deciding whether or not the graph contains a negative weight cycle is in* NL.

Corollary 3.3. *(of Observation 3.1 and Observation 3.2) Algorithm 2 shows that:* Perfect-Matching *(*Decision*) in bipartite planar graphs is in* NL.

3.2 Constructing a Hall-obstacle

In this section, we note the correspondence between the Hall-obstacles in a bipartite planar graph and the negative weight cycles in a related planar graph with suitable weights. Let $G = (A \cup B, E)$ be a bipartite planar graph. Let (G, c, d) be a capacity-demand graph defined as follows: for each vertex $v \in A$, set $d(v) = 1$ and for each vertex $v \in B$, set $d(v) = -1$. For $u \in A, v \in B$, set $c(u, v) = 1$ and $c(v, u) = 0$. Let f' be a pseudo-flow in (G, c, d). Let C^* be a negative weight cycle in $\overleftrightarrow{(G^*)}$ with respect to weights $c - f'$. Let $(V_1 = A_1 \cup B_1, V_2 = A_2 \cup B_2)$ be the directed cut in $\overleftrightarrow{G}$ corresponding to C^*, where V_1 corresponds to the set of faces of $\overleftrightarrow{(G^*)}$ that are in the interior of C^* or equivalently the vertices on $\overleftrightarrow{G}$ that are on one side of the cut corresponding to C^*. Since f' is skew-symmetric, $f'(C^*)$ decomposes into the sum of f's of the faces (in $\overleftrightarrow{(G^*)}$) that are in the interior of C^*. Thus we have:

$$f'(C^*) = |A_1| - |B_1|. \tag{1}$$

Lemma 3.4. *If $(a, b) \in (V_1, V_2)$ such that $a \in A_1, b \in B_2$ and $c(a, b) = 1$ then, moving b from B_2 to B_1 does not increase the weight of the cut (and the corresponding cycle in the dual) with respect to the weights $c - f'$.*

Proof: From Eqn. 1, f' decreases by 1 by such a move; c decreases at least by 1. □

Corollary 3.5. *$\overleftrightarrow{(G^*)}$ has a negative weight cycle with respect to weights $c - f'$ iff (a) there exists one with respect to weights $-f'$, i.e., the total weight contribution from c is zero, and hence iff (b) it has a negative weight cycle with respect to weights $c \cdot n^4 - f'$.*

Theorem 3.6. *(Theorem 1.1 (c))* Hall-Obs *(*Construction*) in a bipartite planar graphs is in* NL.

Proof: Constructing a negative weight cycle with respect to the weights $c \cdot n^4 - f'$ is in NL. The set A_1 forms a Hall-obstacle since $N(A_1) \subseteq B_1$ and $|A_1| > |B_1|$ (see Eqn. 1). □

3.3 Constructing a Perfect Matching

In this section, we describe the Miller and Naor's algorithm (Algorithm 3) for constructing a perfect matching in bipartite planar graphs.

Observation 3.7. *Given a directed graph with polynomially bounded weights w such that there are no negative weight cycles, computing the shortest distance $\text{dist}^w(u, v)$ between any two vertices with respect to weights w is in* NL.

Corollary 3.8. *(of Observation 3.7) Algorithm 3 shows that* Perfect-Matching *(*Decision + Construction*) in bipartite planar graphs is in* NL.

Input : A planar bipartite graph $G = (A \cup B, E)$
Promise: $\overleftrightarrow{(G^*)}$ has no negative weight cycle with respect to $w = c - f'$
Output : A Perfect Matching in G

1 Fix a vertex $s^* \in \overleftrightarrow{(G^*)}$;
2 Set $f''(u^*, v^*) := \text{dist}^w(s^*, v^*) - \text{dist}^w(s^*, u^*)$;
3 Set $f = f'' + f'$;
4 For $u \in A, v \in B$ output "u is matched to v" iff $f(u, v) = 1$;

Algorithm 3: MN-Construction [MN89]

4 The UL Bounds

Suppose we have a directed graph G with polynomially bounded weights on its edges. The weights could be positive or negative. Let s be a fixed vertex in G. Let $d(u, v)$ denote the length of the minimum length path from u to v.

$$V_k := \{v \mid d(s, v) \leq k\}.$$

Let $\text{dist}_k^w(u, v)$ denote the weight of the minimum weight walk (with respect to weights w) of length at most k from u to v. Note that $\text{dist}_k^w(u, v)$ could be negative.

$$\Sigma_k^w := \sum_{v \in V_k} \text{dist}_k^w(s, v).$$

First we describe subroutines adapted from [RA00] that eventually compute $\text{dist}_k^w(s, v)$ and V_k in UL. Next we combine these subroutines with Miller and Naor's algorithm via Weighting Sceheme A in Section 5.2 to obtain the UL bound for perfect matching in bipartite planar graphs. Algorithm 4 and 5 can be combined in a straightforward way (see Appendix 6) to obtain an extention of [RA00] when the graph contains negative weight edges but no negative weight cycle. We call this extension as *Extended-RA Algorithm*. The following lemmas are the consequence of Extended-RA Algorithm and *min-uniqueness* achieved via generalized BTV weights (Weighting Scheme A in Section 5.2).

Lemma 4.1. *Given a directed planar graph with polynomially bounded weights w on its arcs such that there are no negative weight cycles, the shortest distance* $\text{dist}^w(u, v)$ *between any pair of vertices with respect to weights w can be computed in* UL.

Proof: If there are no negative weight cycles then every minimum weight walk between any pair of vertices is a path and $\text{dist}_n^w(u, v) = \text{dist}_\infty^w = \text{dist}^w(u, v)$. In the absence of negative weight cycles, the generalized BTV weights (Section 5.2) guarantee min-uniqueness and hence the Algorithm 4 computes $\text{dist}_n^w(u, v)$ in UL. □

Lemma 4.2. *Given a directed planar graph with polynomially bounded weights w on its arcs, deciding whether or not the graph contains a negative weight cycle is in* coUL.

Proof: Let N denote the sum of the absolute values of the weights on the arcs. If there were no negative weight cycle then $\text{dist}_t^w(u, v)$ remains unchanged for $t \geq n$; otherwise for $t > Nn$ the $\text{dist}_t^w(u, v)$ would decrease for some (u, v). □

Input : (G, s), $(k, |V_k|, \Sigma_k^w)$, and v
Output : $\text{dist}_k^w(s, v)$ ($< \infty$ if $v \in V_k$; ∞ otherwise)

Initialize $c \leftarrow 0$; $s \leftarrow 0$; $\text{dist}_k^w(s, v) \leftarrow \infty$;
foreach $x \in V$ **do**
  Guess a walk of length at most k from s to x;
  if *Guess fails* **then**
    Halt and reject
  else
    Let p be the weight of the walk;
    Set $c = c + 1; s = s + p$;
  end
  if $x = v$ **then** Set $\text{dist}_k^w(s, v) \leftarrow p$
end
if $c = |V_k|$ *and* $s = \Sigma_k^w$ **then**
  Output $\text{dist}_k^w(s, v)$;
else
  Halt and Reject;
end

Algorithm 4: Constructing V_k from $(k, |V_k|, \Sigma_k^w)$ (adapted from [RA00])

Input : (G, s) and $(k, |V_{k-1}|, \Sigma_{k-1}^w)$
Output : $(|V_k|, \Sigma_k^w)$

Initialize $c \leftarrow |V_{k-1}|$; $s \leftarrow \Sigma_{k-1}^w$;
foreach $v \in V$ **do**
  if $v \in V \setminus V_{k-1}$ **then**
    Set $\text{dist}_k^w(s, v) \leftarrow \min_{x:(x,v)\in E(G)}[\text{dist}_{k-1}^w(s, x) + w(x, v)]$;
  end
  if $\text{dist}_k^w(s, v) < \infty$ **then**
    Set $c \leftarrow c + 1$ and $s \leftarrow s + \text{dist}_k^w(s, v)$;
  end
  if *there exist* x_1 *and* x_2 *such that*
  $\text{dist}_{k-1}^w(s, x_1) + w(x_1, v) = \text{dist}_{k-1}^w(s, x_2) + w(x_2, v)$ **then**
    Halt and reject (saying that graph is not min-unique);
  end
end
Set $|V_k| \leftarrow c$ and $\Sigma_k^w \leftarrow s$;
Output $(|V_k|, \Sigma_k^w)$;

Algorithm 5: Computing $(|V_k|, \Sigma_k^w)$ from $(|V_{k-1}|, \Sigma_{k-1}^w)$ (adapted from [RA00])

Input : A directed graph G on n vertices; edge-weights $w : E(G) \to \mathbb{Z}$ such that $|w(e)| \leq n^{O(1)}$; $s, v \in V(G)$; and an integer t
Output : $\text{dist}_t^w(s, v)$

Initialize $V_0 \leftarrow \{s\}$ and $\Sigma_0^w \leftarrow 0$;
for $k = 1$ *to* t **do**
  Compute $(|V_k|, \Sigma_k^w)$ from $(|V_{k-1}|, \Sigma_{k-1}^w)$;
end
Compute $\text{dist}_t^w(s, v)$ from $(|V_t|, \Sigma_t^w)$ and output;

Algorithm 6: Extended-RA Algorithm (adapted from [RA00])

Input : A bipartite planar graph G
Output : A perfect matching in G if one exists; else reject

Construct a capacity-demand graph (G, c, d) as follows: for each vertex $v \in A$, set $d(v) = 1$ and for each vertex $v \in B$, set $d(v) = -1$. For $u \in A, v \in B$, set $c(u, v) = 1$ and $c(v, u) = 0$;
Run Algorithm 1 to construct a pseudo-flow f' in (G, c, d);
Construct a zero-demand graph $(G, c - f')$;
Run Extended-RA Algorithm on $\overleftrightarrow{(G^*)}$ with weights $w = n^4(c - f') + btv$ to compute the shortest distances $\text{dist}_n^w(u, v)$ in $\overleftrightarrow{(G^*)}$, where btv denotes generalized BTV weights (Weighting Scheme A in Section 5.2);
Compute $\text{dist}_n^{c-f'}(u, v)$ in $\overleftrightarrow{(G^*)}$ from the above by ignoring the lower order weights from btv;
Run Algorithm 3 to compute f;
If f is a flow then for $u \in A$ and $v \in B$, output "u is matched to v" $\iff f(u, v) = 1$;
otherwise reject and output "No perfect matching";

Algorithm 7: UL algorithm for Perfect-Matching in bipartite planar graphs

Theorem 4.3. *(Theorem 1.1 (a))* Perfect-Matching*(*Decision + Construction*) in bipartite planar graphs is in* UL.

Proof: The correctness of the above algorithm follows from [MN89]. To see the UL bound, note that the Extended-RA algorithm computes dist_n^w correctly along a unique path assuming min-uniqueness of the weights. If there are no negative weight cycles then the generalized BTV weights (Section 5.2) guarantee min-uniqueness.

Thus: if there are no negative weight cycles in $\overleftrightarrow{(G^*)}$ then we get a valid flow and thus a valid perfect matching along a unique accepting path; otherwise we get f that is not a valid flow and we reject. □

Corollary 4.4. *(Theorem 1.1 (b))* Hall-Obs *(*Decision*) in bipartite planar graphs is in* coUL.

Corollary 4.5. UPM *in bipartite planar graphs is in* UL.

Corollary 4.6. *Single-source, single-sink maximum flow problem in planar networks with polynomially bounded capacities is in* L^{UL}.

Proof: As observed in [MN89], all the subroutines of the above UL algorithm work for any capacity-demand graph such that the sum of the demands is zero. If one knows the value α of the maximum flow then one can construct such a capacity-demand graph by setting demands of all vertices other than s and t to be zero and demand of s to α and demand of t to $-\alpha$. Now, one could search through all possible values of α. □

5 Applications to Related Problems

5.1 Even Perfect Matching in bipartite planar graphs is in NL

Definition 3 (Even-PM)**.** *Given a graph with each edge colored either Red or Blue, an Even-PM is a perfect matching that contains even number of Red edges. Let* Even-PM *denote the problem of deciding whether or not there exists such a perfect matching.*

Theorem 5.1. *(Theorem 1.2 (a))* Even-PM *in bipartite graphs is in* P.

Proof: Given a bipartite graph G, first find a perfect matching M in it. If M is Even-PM we are done, otherwise construct an auxiliary directed graph H with respect to M as follows: $u \rightarrow_H u$ iff $\exists w$ such that $\{u, w\} \in M$ and $\{w, v\} \notin M$. If the matching edge $\{u, w\}$ as well as the non-matching edge $\{w, v\}$ are of the same color then set the weight of the directed edge (u, v) to 0; otherwise set it to 1.

It is easy to check that there exists Even-PM iff H has a cycle of odd weight. Testing for odd weight cycle can be done in NL. It is easy to see that the complexity of the entire procedure is $\mathsf{NL}^{\text{Bip-Perfect-Matching}}$; where Bip-Perfect-Matching denotes the complexity of Perfect-Matching in bipartite graphs. □

Corollary 5.2. *(restatement of Theorem 1.2 (b))* Even-PM *in bipartite planar graphs is in* NL.

5.2 Even-Path in planar DAG is in UL

Definition 4 (Red-Blue-Path)**.** *Given a directed graph with each edge colored either Red or Blue, a* Red-Blue-Path *from s to t is a (simple) directed path from s to t such that the two consecutive edges are of different colors. The* Red-Blue-Path *problem is the problem of deciding whether or not there is a Red-Blue-Path from s to t.*

Definition 5 (Even-Path)**.** *Given a directed graph and two notes s and t, an* Even-Path *from s to t is a (simple) directed path from s to t containing even number of edges. The* Even-Path *problem is the problem of deciding whether or not there is an Even-Path from s to t.*

Theorem 5.3 ([Kul09])**.** Red-Blue-Path *in planar DAG is* NL*-complete.*

In this section, we prove that the Even-Path problem (which can be viewed as a relaxation of the Red-Blue-Path problem as a path starting with say Red edge and ending with say Blue edge is always of even length) in planar DAG is in fact in UL. Our proof involves a combination of two different isolation techniques that are currently available.

The weighting scheme A

Weighting scheme A is a generalization of the weight function in [BTV09] to planar graphs. In other words, given a directed planar graph G, we construct a log-space computable edge weight function with respect to which any simple cycle in G has non-zero weight. Tewari and Vinodchandran [TV10] give a log-space construction of such a weight function by an application of Green's Theorem.

Input : A planar graph G
Output : An edge weight function w_A such that for any simple cycle C in G $w_A(C) \neq 0$

1 Compute a spanning tree T in G;
2 For any arc $e \in \overleftrightarrow{T}$, set $w_A(e) = 0$;
3 Let R denote the spanning tree in G^* consisting of the edges that do not belong to T. Fix a root r for R (say the unbounded face) and let $\overrightarrow{R}$ denote the orientation of R where each edge is oriented towards the root;
4 An arc $e^* = (u, v) \in \overrightarrow{R}$ separates the tree R into two subtrees. Let α_u denote the number of vertices in the subtree containing u. Set $w_A(u, v) = \alpha_u$ and $w_A(v, u) = -\alpha_u$;
5 Set $w_A(e) = w_A(e^*)$ for every $e \in E(G)$ where e^* is the (directed) dual edge of e;

Algorithm 8: Weighting Scheme A

Lemma 5.4 (adaptation of [BTV09])**.** *With respect to the weight function w_A the absolute value of the sum of the weights of the arcs along any directed cycle is equal to the number of faces in the interior of the cycle.*

Proof: For a simple cycle C of G, let us define the weight of C, $w(C)$, to be the sum of weights of the edges lying along C in clockwise order. It suffices to show that for a facial

cycle F of G, $w(F) = +1$. This is because for a simple cycle C:

$$w(C) = \sum_{F \in \text{Interior}(C)} w(F).$$

But $w(F)$ equals the sum of the weights of dual edges (in G^*) outgoing from the dual vertex $F^* \in V(G^*)$, so it suffices to show that for every vertex $u \in V(G^*)$:

$$\sum_{v:(u,v)\in E(G^*)} \alpha_v = +1.$$

Observe that the number of nodes in the subtree rooted at u is one more than sum of the number of vertices in the subtrees rooted at v for various v, such that (u, v) is a dual edge. This, together with the skew symmetry of the weights $w_A(u, v)$, completes the proof. □

Lemma 5.5. *Let G be a planar DAG and u and v be any two vertices in G. Then with respect to the weight function w_A, (a) if P_1 and P_2 are two minimum weight Even-Paths from u to v, then $P_1 \oplus P_2$ divides the plane into at most two bounded regions; (b) no three minimum weight Even-Paths from u to v share a common vertex w other than u and v, such that the path segments between the vertices u and w and between w and v are not identical. (c) there are at most $2n^4$ minimum weight Even-Paths from u to v.*

Proof. (a) For the sake of contradiction let C_1, C_2 and C_3 be any three bounded regions of $P_1 \oplus P_2$. Let P_{ij} be the restriction of the i-th path to the j-th for $i \in \{1, 2\}$ and $j \in \{1, 2, 3\}$. Observe that $w_A(P_{1j}) \neq w_A(P_{2j})$ since C_j is a simple cycle and by Lemma 5.4 we have that $w_A(C_j) \neq 0$. Now the parity of the lengths of the path segments $P_{1,j}$ and $P_{2,j}$ are different since if they were the same, we could replace the higher weighted segment with the lower weighted one and get an even length path of lesser weight. This implies that $w_A(C_1+C_2+C_3)$ is odd since the weight of each C_i is odd. Let $P'_i = \bigcup_j P_{ij}$ for $i \in \{1, 2\}$. Therefore either $w_A(P'_1)$ is odd or $w_A(P'_2)$, but not both. Without loss of generality lets assume $w_A(P'_1)$ is odd. For each j pick the path segment between P_{1j} and P_{2j} that has lesser weight to create a set say P'. Now $w_A(P')$ is strictly smaller than both $w_A(P'_1)$ and $w_A(P'_2)$. If $w_A(P')$ is odd then replace P'_1 with P' and if $w_A(P')$ is even then replace P'_2 with P' to get a path of smaller weight and same parity. This is a contradiction. Thus $P_1 \oplus P_2$ has at most two bounded regions.

(b) Let P_1, P_2 and P_3 be three minimum weight paths from u to v that share a common vertex (say w) such that the segments of each of the three paths between the vertices u and w and between w and v are distinct. In other words, if P'_i and P''_i are the segments of P_i between the vertices u and w and between w and v respectively (for $i \in \{1, 2, 3\}$), then $\{P'_i\}$ are pairwise non-identical and so are $\{P''_i\}$. There exists at least two path segments between P'_1, P'_2 and P'_3 whose lengths have the same parity. Without loss of generality assume its P'_1 and P'_2. Now if $w_A(P'_1) \neq w_A(P'_2)$ then since they have the same parity we can pick the lesser weight path between P'_1 and P'_2 and similarly the lesser weight path between P''_1 and P''_2 and append them to get an even path of weight less than either that of P_1 or P_2 from u to v. Thus we can assume $w_A(P'_1) = w_A(P'_2)$. By Lemma 5.4, this implies that $P'_1 \oplus P'_2$ as at least two bounded regions. Moreover since P''_1 and P''_2 are also not identical, therefore $P''_1 \oplus P''_2$ has at least one one bounded region. Thus $P_1 \oplus P_2$ has at least 3 bounded regions, thus contradicting part (a).

(c) Let a, b, c and d be four vertices in G and let $\mathcal{P}_{a,b,c,d}$ be the set of all minimum weight even length paths from u to v that pass through the vertices a, b, c and d in that order and are vertex disjoint between the vertices a and b and between the vertices c and d respectively. Then by part (b), $\mathcal{P}_{a,b,c,d}$ will have at most 2 paths. Since the total number of such tuples is at most n^4, therefore the number of minimum weight, even length u-v paths is bounded by $2n^4$. □

Constructing an auxiliary graph

Construct a directed (multi)graph G' from G as follows: the vertex set of G' is the vertex set of G. An edge (v_i, v_j) is in G' if and only if there exists a vertex v_k in G and the edges (v_i, v_k) and (v_k, v_j) are in G. The weight w of an edges in G' is the sum of the weights of the corresponding two edges in G.

Now Lemma 5.6 follows by definition of G' and part (c) of Lemma 5.5.

Lemma 5.6. *(a) G has an Even-Path from u to v if and only if G' has a directed path from u to v; (b) the number of minimum weights paths from u to v in G' with respect to w_A is at most $2n^4$.*

Weighting scheme B

Our weighting scheme B is based on a well known hashing scheme based on primes, due to Fredman, Komlós and Szemerédi [FKS84].

Lemma 5.7 ([FKS84])**.** *Let c be a constant and S be a set of n-bit integers with $|S| \leq n^c$. Then there is a c' and a $c' \log n$-bit prime number p so that for any $x \neq y \in S$ $x \not\equiv y \pmod{p}$.*

Hoang used this scheme to give better upper bounds for Perfect-Matching in certain classes of graphs [Hoa10]. Pavan, Tewari and Vinodchandran showed that reachability in graphs where the number of paths from s to any vertex is bounded by a polynomial is in UL, by applying this hashing scheme. We use Lemma 5.7 here to define a weight function with respect to which G' in min-unique.

Let p_i be the i^{th} prime number. Consider the lexicographical ordering of the edges of G' and denote the j^{th} edge in this ordering by e_j. Define the i^{th} weight function (for $1 \leq i \leq q(n)$ and an appropriate polynomial $q(n)$ dictated by Lemma 5.7), $w_{B_i}(e_j) = 2^j(\mod p_i)$.

Lemma 5.8 (Adapted from [PTV10])**.** *There exists an $i \leq q(n)$ such that the graph G' with respect to the weight function $W_i = w_A \cdot n^{10} + w_{B_i}$ is min-unique.*

Proof. Let $\mathcal{P}_v$ be the set of minimum weight paths from s to a vertex v in G', with respect to w_A. Then by Lemma 5.6, $|\mathcal{P}_v|$ is bounded by $2n^4$. It follows from Lemma 5.7 that with respect to some w_{B_i}, all paths in $\bigcup_v \mathcal{P}_v$ will have distinct weights. Therefore G' is min-unique with respect to W_i for some i. □

For each $i \in [q(n)]$, check if G' is min-unique with respect to W_i or not. Once we have an appropriate i, we can decide reachability in G' in UL [RA00]. By Lemma 5.6 a path in G' corresponds to an EvenPath in G and thus we have Theorem 5.9.

Theorem 5.9. *(Theorem 1.3)* Even-Path *in planar DAG is in* UL.

5.3 Neg-Cycle Problem

Lemma 5.10. *In planar graphs:*

(a) Bip-Perfect-Matching $\leq$ Neg-Cycle $\leq$ Min-Wt-PM *(bipartite), and*

(b) Neg-Cycle(Decision + Construction) *is in* $\mathsf{NL} \cap \mathsf{SPL}$.

Proof Sketch: The first inequality in Part (a) follows from the Miller and Naor's algorithm. To prove the second inequality in Part (a), given a directed graph G with polynomially bounded weights, construct an undirected graph H as follows: for each vertex $u \in G$ we have two vertices u_{in} and u_{out} in H; a directed edge (u, v) in G becomes an undirected edge between u_{out} and v_{in} in H. In addition we have weight 0 edges between u_{in} and u_{out}. It is easy to check that G has a negative weight cycle iff H has a perfect matching of negative weight. Note that H is bipartite. Moreover, by replacing a vertex of degree d by a cycle of length d we can transform G such that the sum of in-degree and out-degree at any vertex is at most 3. Now H will also be planar. □

6 Open Ends

1. Is Neg-Cycle (Decision) in planar graphs in UL?
2. Is Odd-Cycle in planar graphs in $\oplus$L?
3. Is Perfect-Matching (Decision) in bipartite planar graphs in coUL?
4. Is Min-Wt-PM in bipartite planar graphs in NL?
5. Is Max-Matching in bipartite planar graphs in NL?

Acknowledgement

We would like to thank Prajakta Nimbhorkar for discussion in the initial stages of the work, in particular for pointing out that the Neg-Cycle problem is in NL. We would like to thank N. V. Vinodchandran for pointing out references [FKS84] and [PTV10] which are crucially used in the proof of Theorem 5.9.

References

[ARZ99] Eric Allender, Klaus Reinhardt, and Shiyu Zhou. Isolation, matching, and counting: Uniform and nonuniform upper bounds. *Journal of Computer and System Sciences*, 59:164–181, 1999.

[BTV09] Chris Bourke, Raghunath Tewari, and N. V. Vinodchandran. Directed planar reachability is in unambiguous log-space. *ACM Trans. Comput. Theory*, 1(1):1–17, 2009.

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

[DKLM10] Samir Datta, Raghav Kulkarni, Nutan Limaye, and Meena Mahajan. Planarity, determinants, permanents, and (unique) matchings. *ACM Trans. Comput. Theory*, 1(3):1–20, 2010.

[DKR10] Samir Datta, Raghav Kulkarni, and Sambuddha Roy. Deterministically isolating a perfect matching in bipartite planar graphs. *Theory of Computing Systems*, 47:737–757, 2010. 10.1007/s00224-009-9204-8.

[DKTV10] Samir Datta, Raghav Kulkarni, Raghunath Tewari, and N. V. Vinodchandran. Space complexity of perfect matching in bounded genus bipartite graphs. Technical Report TR10-079, Electronic Colloquium on Computational Complexity, 2010.

[Edm65] J. Edmonds. Paths, trees and flowers. *Canad. J. Math.*, 17:449–467, 1965.

[FKS84] Michael L. Fredman, János Komlós, and Endre Szemerédi. Storing a sparse table with 0(1) worst case access time. *J. ACM*, 31:538–544, June 1984.

[Hoa10] Thanh Minh Hoang. On the matching problem for special graph classes. In *IEEE Conference on Computational Complexity*, pages 139–150, 2010.

[Kas67] P. W. Kasteleyn. Graph theory and crystal physics. *Graph Theory and Theoretical Physics*, 1:43–110, 1967.

[KMV08] Raghav Kulkarni, Meena Mahajan, and Kasturi R. Varadarajan. Some perfect matchings and perfect half-integral matchings in NC. *Chicago Journal of Theoretical Computer Science*, 2008(4), September 2008.

[Kul09] Raghav Kulkarni. On the power of isolation in planar graphs. Technical Report TR09-024, Electronic Colloquium on Computational Complexity, 2009.

[LP83] Andrea Lapaugh and Christos Papadimitriou. The even-path problem for graphs and digraphs. *Networks Volume 14, Issue 4 , Pages 507 - 513*, 1983.

[LP86] L. Lovász and M.D. Plummer. *Matching Theory*, volume 29. North-Holland Publishing Co, 1986.

[MN89] G.L. Miller and J. Naor. Flow in planar graphs with multiple sources and sinks. In *Foundations of Computer Science, 1989., 30th Annual Symposium on*, pages 112 –117, 1989.

[MV00] Meena Mahajan and Kasturi R. Varadarajan. A new NC-algorithm for finding a perfect matching in bipartite planar and small genus graphs. In *ACM Symposium on Theory of Computing*, 2000.

[MVV87] Ketan Mulmuley, Umesh Vazirani, and Vijay Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7:105–113, 1987.

[Ned99] Zhivko Prodanov Nedev. Finding an even simple path in a directed planar graph. *SIAM Journal on Computing, Volume 29 , Issue 2, Oct 99, 685-695*, 1999.

[PTV10] Aduri Pavan, Raghunath Tewari, and N. V. Vinodchandran. On the power of unambiguity in logspace. Technical Report TR10-009, Electronic Colloquium on Computational Complexity, 2010.

[PY82] Christos H. Papadimitriou and Mihalis Yannakakis. The complexity of restricted spanning tree problems. *J. ACM*, 29(2):285–309, 1982.

[RA00] Klaus Reinhardt and Eric Allender. Making nondeterminism unambiguous. *SIAM Journal of Computing*, 29:1118–1131, 2000. An earlier version appeared in FOCS 1997, pp. 244–253.

[TV10] Raghunath Tewari and N. V. Vinodchandran. Green's theorem and isolation in planar graphs. Technical Report TR10-151, Electronic Colloquium on Computational Complexity, 2010.

[Val79] Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.

[Vaz88] Vijay Vazirani. NC algorithms for computing the number of perfect matchings in $k_{3,3}$–free graphs and related problems. In *Proceedings of SWAT '88*, pages 233–242, 1988.

[Vol99] Heribert Vollmer. *Introduction to Circuit Complexity - A Uniform Approach.* Springer-Verlag, 1999.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de