Theory Comput Syst (2010) 47: 737–757
DOI 10.1007/s00224-009-9204-8

Deterministically Isolating a Perfect Matching in Bipartite Planar Graphs

Samir Datta · Raghav Kulkarni · Sambuddha Roy

Published online: 17 March 2009

Abstract We present a deterministic Logspace procedure, which, given a bipartite planar graph on n vertices, assigns $O(\log n)$ bits long weights to its edges so that the minimum weight perfect matching in the graph becomes unique. The *Isolation Lemma* as described in Mulmuley et al. (Combinatorica 7(1):105–131, 1987) achieves the same for general graphs using randomness, whereas we can do it deterministically when restricted to bipartite planar graphs. As a consequence, we reduce both decision and construction versions of the perfect matching problem in bipartite planar graphs to testing whether a matrix is singular, under the promise that its determinant is 0 or 1, thus obtaining a highly parallel SPL algorithm for both decision and construction versions of the bipartite perfect matching problem. This improves the earlier known bounds of non-uniform SPL by Allender et al. (J. Comput. Syst. Sci. 59(2):164–181, 1999) and NC^2 by Miller and Naor (SIAM J. Comput. 24:1002–1017, 1995), and by Mahajan and Varadarajan (Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing (STOC), pp. 351–357, 2000). It also rekindles the hope of obtaining a deterministic parallel algorithm for constructing a perfect matching in non-bipartite planar graphs, which has been open for a long time. Further we try to find the lower bound on the number of bits needed for de-

This work was done while the second author was visiting Chennai Mathematical Institute.

S. Datta
Chennai Mathematical Institute, Chennai, India
e-mail: sdatta@cmi.ac.in

R. Kulkarni (✉)
University of Chicago, Chicago, USA
e-mail: raghav@cs.uchicago.edu

S. Roy
IBM Research Laboratory, New Delhi, India
e-mail: sambuddha@in.ibm.com

terministically isolating a perfect matching. We show that our particular method for isolation will require $\Omega(\log n)$ bits. Our techniques are elementary.

Keywords Isolation lemma · Deterministic isolation · Perfect matching · Planar graphs · NC

1 Introduction

The *Matching Problem* is one of the most well-studied in the field of parallel complexity. Attempts to solve this problem have led to the development of a variety of combinatorial, algebraic and probabilistic tools which have found applications even outside the field of parallel complexity. Since the problem is still open, researchers linger around it in search of new techniques, if not to solve it in its whole generality, then at least under various natural restrictions. In this paper, we will focus on the deterministic complexity of the *Matching Problem* under its planar restrictions.

1.1 The Matching Problem

Definition 1.1 A matching in an undirected graph is a set of vertex disjoint edges.

Such a collection of edges is called "independent" and the vertices which are end points of an edge in the matching are called "matched". See [14] for an excellent survey on matchings.

A computational question one can ask here is, given a graph, to find a matching of the maximum cardinality.

Definition 1.2 A perfect matching in a graph is a matching in which every vertex is matched.

One may ask various computational questions about perfect matchings in graphs. We will consider the following three questions:

Question 1 (Decision) Is there a perfect matching in a given graph?
Question 2 (Search) Construct a perfect matching in a graph, if it exists.
Question 3 (Uniqueness Testing or UPM) Does a given graph have exactly one perfect matching?

There are polynomial time algorithms for the above graph matching problems and historically people have been interested in studying the parallel complexity of all the three questions above. The UPM question for bipartite graphs is deterministically parallelizable, i.e., it lies in the complexity class NC [13]. (See any standard complexity text for a formal definition, say [20]). Intuitively, NC is the complexity class consisting of problems having a parallel algorithm which runs in polylogarithmic time using polynomially many processors which have access to a common memory.

It is the class consisting of so called "well parallelizable" problems. NC is inside P-problems having a sequential polynomial time algorithm. Whether the *Matching Problem* is deterministically parallelizable remains a major open question in parallel complexity.

Open Problem 1.3 Is Matching in NC?

The best we know till now is that *Matching* is in *Randomized* NC. See for example, [12, 18]. Several restrictions of the matching problem are known to be in NC, for example, bipartite planar graphs [15, 17], graphs with polynomially bounded number of perfect matchings [8] etc. Whether the search version reduces to the decision version in parallel settings has also not been answered yet.

1.2 The (Randomized) Isolation Lemma

Lemma 1.4 [18] *Let* $[m]$ *be the set* $\{1, 2, \ldots, m\}$ *and* F *be a family of subsets of* $[m]$. *If one assigns weights to each element of* $[m]$ *uniformly and independently at random from* 1 *to* $2m$, *then with high* ($> \frac{1}{2}$) *probability, the minimum weight subset of* F *will be unique*. (*Weight of a subset is the sum of the weights of the elements in it*.)

Using the Isolation Lemma, [18] obtained a simple *Randomized* NC algorithm for finding a perfect matching in arbitrary graphs.

1.3 Matching in non-uniform SPL

Building on the Isolating Lemma, Allender et al. [3] proved a non-uniform bound for matching problem which allows us to replace the randomization by a polynomial length advice string. Hence, we know that matching is parallelizable with a polynomial length advice.

Definition 1.5 SPL is a promise class that is characterized by the problem of checking whether a matrix is singular under the promise that its determinant is either 0 or 1.

SPL is inside $\oplus$L and inside $\oplus_p$L for all p. While UL (unambiguous Logspace) is inside SPL. With current knowledge, NL (nondeterministic Logspace) is believed to be incomparable with SPL. Both NL and SPL are known to lie inside NC^2.

Definition 1.6 An integer valued function is said to be in GapL/poly if for every positive integer n there exists an advice string A_n such that:

- length of A_n is polynomially bounded in n
- once A_n is given, the value of the function on any input of size n can be computed in GapL.

Definition 1.7 A language L is in *non-uniform* SPL, if its characteristic function χ_L is in GapL/poly.

Theorem 1.8 [3] *Matching is in* non-uniform SPL.

Reference [3] also showed that under a plausible assumption of secure pseudorandom generators, *Matching* is in SPL.

1.4 Matchings in Planar Graphs and Deterministic Isolation

The situation for planar graphs is interesting because of the fact that counting the number of perfect matchings in planar graph is in NC [10, 19] whereas constructing one perfect matching is not yet known to be parallelizable. However, for bipartite planar graphs, people have found NC algorithms [15, 17].

The Isolation Lemma crucially uses randomness in order to isolate a minimum weight set in an arbitrary set system. It is conceivable, however, to exploit some additional structure in the set system to eliminate this randomness. Indeed, recently [5] building upon a technique from [2] were able to isolate a directed path in a planar graph by assigning small *deterministic* weights to the edges. The lemma that sits at the heart of that result says that there is a simple deterministic way to assign weights so that each directed cycle (in a grid graph) gets a non-zero weight. This is shown to imply that if two paths get the same weight neither of them is a min-weight path.

The authors of [12] observe that reachability in directed graphs reduces to bipartite perfect matching. [6] observe that directed planar reachability reduces to planar bipartite perfect matching. [5] find an isolating weighting scheme for paths in directed planar graphs. Motivated by their result we explore the possibility of such an isolation for perfect matchings in planar graphs. Given the facts from [12] and [6] it is natural to ask whether the weighting scheme from [5] generalizes for bipartite planar perfect matchings and we prove exactly the same in this paper.

Our attempt is to assign weights so that the alternating sum is non-zero for each alternating cycle—here alternating sum is the signed sum of weights where the sign is opposite for successive edges. Since alternating cycles result from the superimposition of two matchings, we are able to isolate a min-weight matching.

Therefore, we are able to devise an NC algorithm for bipartite planar graphs which is conceptually simple, different from the other known algorithms and tightens its complexity to the smaller class SPL. The search problem for matching in non-bipartite planar graphs still remains open even though the corresponding decision and counting versions are known to be in NC. Our algorithm rekindles the hope for solving general planar matching in NC by generalizing our weighting scheme from bipartite planar graphs to non-bipartite planar graphs.

In the context of our weighting scheme and that of [5], one natural question is what is the minimum size of the weights. We show that any weighting that follows our method must use $\Omega(\log n)$ bit long weights. The same holds for weighting scheme from [5].

1.5 Organization

In Sect. 2 we describe the basic definitions and facts used in this paper. In Sect. 3 we prove our main result that there exists a Logspace procedure which assigns $O(\log n)$ bit weights to the edges of a bipartite planar graph so that the minimum weight perfect matching becomes unique. Section 4 considers further restrictions and their complexities. The motivation for this section is to identify a natural subclass of planar graphs in which matching problems are as easy as Logspace. Indeed, we show that UPM question for outerplanar graphs is solvable in Logspace. Section 5 discusses the possibility of generalizing our result in various directions. It also contains the lower bound on the size of the weights obtained using our weighting scheme.

2 Preliminaries

Here we describe the technical tools that we need in the rest of the paper. Refer to any standard text (e.g. [20]) for definitions of the complexity classes $\oplus\mathsf{L}$, $\oplus_p\mathsf{L}$, NL, UL, NC^2. For graph-theoretic concepts, for instance, *planar graph*, *outerplanar graph*, *spanning trees*, *adjacency matrix*, *Laplacian matrix* of a graph, we refer the reader to any standard text in graph theory (e.g. [7]).

2.1 Definitions and Facts

We will view an $n \times n$ grid as a graph simply by putting the nodes at the grid points and letting the grid edges act as the edges of the graph.

Definition 2.1 *Grid graphs* are simply subgraphs of an $n \times n$ grid for some n. See Fig. 1 for an example.

Definition 2.2 (Block) We call each unit square of the grid a *block*, i.e., block is simply the subgraph of grid corresponding to a cycle of length 4.

Definition 2.3 We call a graph an *almost grid graph* if it consists of a grid graph and possibly some additional diagonal edges which all lie in some single row of the grid. Moreover all the diagonal edges are parallel to each other. See Fig. 2.

In this paper we will consider weighted grid graphs where each edge is assigned a rational weight.

Definition 2.4 (Signs of Edges and Blocks of the Grid)

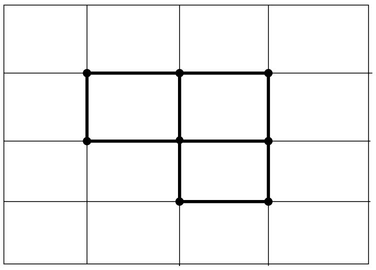
Fig. 1 A grid graph

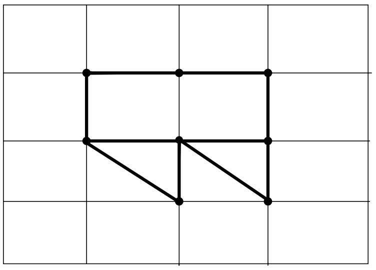
Fig. 2 An almost grid graph

(1) Given a grid, assign a "+" sign to all the vertical edges and a "−" sign to all the horizontal edges.
(2) Assign a sign of $(-1)^{i+j}$ to the block in the ith row and jth column (adjacent blocks get opposite signs).

Definition 2.5 (Circulation of a Block) Given a weighted grid graph G, the *circulation* of a block B(denoted $\mathrm{circ}(B)$) in G is the signed sum of weights of the edges of it: $\mathrm{circ}(B) = \sum_{e\in B}\mathrm{sign}(e)\mathrm{weight}(e)$.

Definition 2.6 (Circulation of a Cycle) Given a weighted grid graph G and a cycle $C = (e_0, e_1, \ldots, e_{2k-1})$ in it, where e_0 is, say, the leftmost topmost vertical edge of C; we define the circulation of a cycle C as $\mathrm{circ}(C) = \sum_{i=0}^{2k-1}(-1)^i\mathrm{weight}(e_i)$.

The following lemma plays a crucial role in constructing non-vanishing circulations in grid graphs as will be described in the next section.

Lemma 2.7 (The Block Decomposition of Circulations) *The circulation of a cycle C in a grid graph G is equal to the signed sum of the circulations of the blocks of the grid which lie in the* interior *of C, in absolute value*:

$$|\mathrm{circ}(C)| = \left|\sum_{B\in\mathrm{interior}(C)}\mathrm{sign}(B)\mathrm{circ}(B)\right|$$

Proof Consider the summation on the right hand side. The weight of any edge in the *interior* of C will get canceled in the summation because that edge will occur in exactly two blocks which are adjacent and hence with opposite signs. Now what remains are the boundary edges. Call two boundary edges *adjacent* if they appear consecutively on the cycle C. □

Claim 2.8 *Adjacent boundary edges get opposite signs in the summation on the right hand side above.*

Proof We have to consider two cases, either the adjacent boundary edges lie on adjacent blocks, in which case since adjacent blocks have opposite signs, these edges will also get opposite signs as they are both vertical or horizontal edges. See Fig. 3. In the other case, when adjacent boundary edges do not lie on adjacent blocks, there are two subcases. In first subcase, they lie on two different blocks which are diagonally next to each other. In this case, both blocks will have the same sign but since one edge is vertical and the other is horizontal, the effective sign of the edges will be opposite. In the second subcase when the two adjacent boundary edges lie on the same block the claim is trivial. See Fig. 3. Hence, the adjacent boundary edges will get opposite sign in the summation. This completes the proof that the right hand side summation is precisely $+\mathrm{circ}(C)$ or $-\mathrm{circ}(C)$. □

We will also have occasion to employ the following lemma and we record it here:

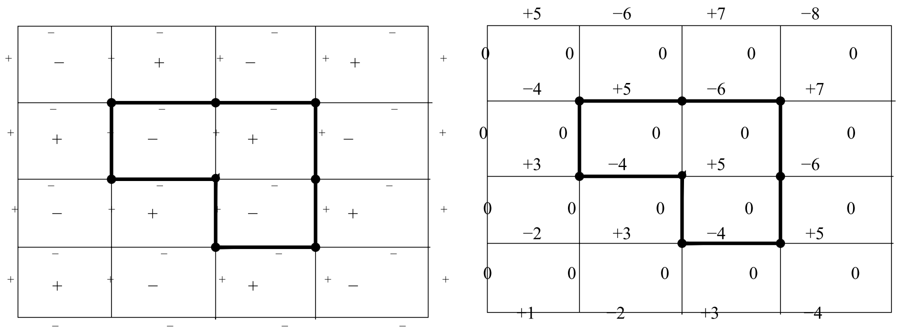

Fig. 3 Signs and weights of the blocks and the edges of a grid

Lemma 2.9 (Temperley's Bijection) *The spanning trees of a planar graph are in one to one correspondence with perfect matchings in a bipartite planar graph. Moreover the correspondence is weight preserving.*

This bijection was first observed by Temperley around 1967. Recently [11] have found a *Generalized Temperley Bijection* which gives a one-to-one weight preserving mapping between directed rooted spanning trees or arborescences in a directed planar graph and perfect matchings in an associated bipartite planar graph.

2.2 Planar Matching and Grid Graphs

Grid graphs have turned out to be useful for solving the reachability question in directed planar graphs, cf. [1, 5]. Motivated by this fact we explore the possibility of reducing planar matching problem to that of grid graphs. Non-bipartiteness becomes an obstacle here which leaves us with the following observations:

Lemma 2.10 *One can convert, in Logspace, any bipartite planar graph into a grid graph such that the perfect matchings remain in one-to-one correspondence.*

Proof This is described in [6]. It follows closely the procedure for embedding a planar graph into a grid, described by [1]. □

Though non-bipartiteness is an issue, we can get rid of it to a certain extent, though as expected, not completely.

Lemma 2.11 *Any planar graph, not necessarily bipartite, can be converted, in Logspace, to an* almost grid graph *while maintaining the one to one correspondence between the perfect matchings.*

Proof This procedure is analogous to the previous one except that we can observe that the edges which are causing non bipartiteness can be elongated into a long path and placed in a grid so that only in a single row one needs to use a diagonal edge. Follow the procedure of [2]. All non-tree edges can be made to intersect a row in below all the tree edges. One can accommodate non-bipartiteness using this row. □

3 Bipartite Planar Perfect Matching in SPL

In this section, we will give a simple algorithm for finding a perfect matching in bipartite planar graphs, also improving over its complexity by putting it in SPL. Earlier the best known bound was NC^2. See for example [15, 17] and of non-uniform SPL from [3]. At the core of our algorithm, lies a procedure to deterministically assign the small (logarithmic bit long) weights to the edges of a bipartite planar graph, so that the minimum weight perfect matching becomes unique. A simple observation about non-vanishing circulations in bipartite planar graphs makes it possible to isolate a perfect matching in the graph, which can be further extracted out using an SPL query.

3.1 Non-Vanishing Circulations in Grid Graphs

We are interested in assigning the small weights to the edges of a grid so that any cycle in it will have non-zero circulation. This weighting scheme is at the heart of the isolation of perfect matchings in grid graphs. The procedure runs in Logspace.

The key observation that our weighting scheme uses is the following: It suffices to assign the weights such that

(1) Circulation of every block is non-zero.
(2) Circulations of adjacent blocks are opposite in sign.

Weighting Scheme.

(1) Weight of any vertical edge is zero.
(2) Weight of ith (from left) horizontal edge in jth (from bottom) horizontal row is $(-1)^{i+j}(i+j-1)$.

Lemma 3.1 *One can assign, in Logspace, small ($O(\log n)$ bits) weights to the edges of a grid so that circulation of any cycle becomes non-zero. (One weighting scheme which guarantees non-zero circulation for every cycle in the grid is shown in Fig. 3.)*

Proof We assign all vertical edges weight 0 and horizontal edges are assigned weights as shown in Fig. 3 and defined explicitly by the formula in above weighting scheme. The weighting scheme makes sure that the circulation of any block is either $+1$ or -1. Moreover, the circulation of a block is positive if and only if its sign is positive. Now, using the Block Decomposition of Circulations (Lemma 2.7), we have that the circulation of any cycle, in absolute value, is precisely the number of blocks in the interior of it, and hence is never zero. □

3.2 Non-Vanishing Circulations: A Direct Method

The procedure of achieving non-vanishing circulations for all cycles in bipartite planar graphs can be made more abstract. One does not need to convert the bipartite planar graph into a grid graph. Instead, it suffices to add spurious edges to it so as to make it an Eulerian bipartite planar graph. The block decomposition of circulation holds in any Eulerian bipartite planar graph and one can describe an alternate procedure for coming up with the weighting scheme that achieve non-vanishing circulations. This abstract procedure is analogous to the procedure for assigning Pffafian orientation to planar graphs [10]. This gives a hope of generalizing our method for any Pffafian graph which we leave as an open possibility. As we will see in later section, non-vanishing circulations for bipartite planar graphs would imply the weighting scheme used by [5] to prove planar reachability is in Unambiguous Logspace. This means that if one generalizes our method for $K_{3,3}$-free bipartite graphs then it would also prove the reachability in $K_{3,3}$-free graphs is in UL. Keeping these implications in mind, we try to abstract out our procedure without explicitly having to go through grid graphs.

Step 1: *The procedure to transform a bipartite planar graph into an Eulerian bipartite planar graph:*
Perform an Euler traversal on a spanning tree in the dual graph. While performing the traversal, suppose (u, v) is the edge in the original graph through which you enter from one face to another. Make sure that when you leave the first face, all the vertices in the face, except for the end points u and v of the edge through which you go to the next face, are of even degree. To guarantee this one can do the following:

- *Step 1a* Start with one end point say u of the edge (u, v) through which we go to the next face. Visit all the vertices of the face in a cyclic ordering, every time connect an odd degree vertex to the next vertex by a spurious edge.
- *Step 1b* If the next vertex is also of odd degree then go to its next vertex and connect both these vertices to the next vertex of even degree in the order or to v whichever comes first. If the next vertex is of even degree then we have pushed the oddness one step further.
- *Step 1c* Continue the same procedure till we either remove all the oddness from the face or push the oddness eventually to v.
- *Step 1d* In the process, the graph might become a multi-graph, i.e., between two nodes we may have multiple edges, but this can be taken care of by replacing every multi-edge by a path of length 3. The bipartiteness is preserved in the process. Moreover at the moment when you leave a face through and edge (u, v) all the vertices, except possibly u and v, of the previous faces visited will be of even degree. Eventually when the Euler traversal on the dual tree returns to the root through an edge (u, v), all but possibly u and v have odd degree. If both u and v have odd degree then add a path of length 3 between u and v.

Step 2: *Fix the signs for faces and the edges:*
After Step 1, the graph has become Eulerian, and hence the dual graph is bipartite.

- *Step 2a Assign alternating signs to adjacent faces:* Form a bipartition of the dual, and fix all the faces in one bipartition to have $+$ sign and the others to have $-$ sign. Any two adjacent faces will have opposite signs. Here, faces will act as blocks.
- *Step 2b Assign alternating signs to adjacent edges of every face:* Consider an auxiliary graph obtained from the bipartite planar graph as follows: Every new vertex corresponds to an edge in the graph. Join two new vertices by a new edge iff the corresponding edges in the original graph are adjacent along some face. (Reader may confuse this auxiliary graph with Line Graph but one can easily see that the

auxiliary graph constructed is planar whereas the Line Graph would not be planar if we have a vertex of degree 6.) Now since both the original graph and its dual are bipartite, the auxiliary graph will also be bipartite and hence edges in the two bipartitions get opposite signs ensuring that around every face the signs are alternating.

Step 3: *Assign small weights to the edges to get non-vanishing circulation:*
Now make another Euler traversal on the dual tree everytime assigning the weight to the dual tree edge through which you traverse to the next face so that the circulation of the face you leave is exactly same as the sign of the face. All non-tree edges will be assigned zero weight. It is easy to see that all the weights assigned are small ($O(\log n)$ bits).

The Decomposition of Circulations Again, we claim that the circulation of a cycle will decompose into signed sum of circulations of the faces in the interior of it. Signs of face are defined by the above procedure. Note that adjacent faces get opposite signs. To make sense of the definition of circulation of a face, note that the above procedure assigns signs to the edges of the graph so that adjacent edges of a face get opposite signs. Circulation of a face is defined as the signed sum of the weights of its edges. It is easy to see that for any cycle in the graph, if we take the signed sum of circulations of the faces in the interior of it, then the weight of every edge in the interior gets counted once positive and once negative. Thus, only non-zero contribution comes from the boundary edges. We leave it to the reader to check that the boundary edges contribute exactly the circulation of the cycle. The idea is the following. Consider two consecutive edges say e_1 and e_2 of a cycle and consider the signs s_1 and s_2 of these edges assigned by the above procedure, and let f_1 and f_2 be the faces in the interior of the cycle which contain e_1 and e_2 respectively (possibly $f_1 = f_2$). If $s_1 = s_2$ then signs of f_1 and f_2 must be opposite. If s_1 and s_2 are opposite then signs of f_1 and f_2 must be the same. The crucial property used is that signs of the edges are such that along any face the signs of adjacent edges are opposite. Thus, suppose e_1 and e_2 share an endpoint v and $s_1 = s_2$, then number of edges in the interior of the cycle which are also incident on v must be odd. Thus the faces f_1 and f_2 must have opposite signs. The other case is symmetric.

The weights assigned by the above procedure are such that the circulation of a face is $+1$ if the face has positive sign otherwise it the circulation is -1. Thus, for any cycle, the circulation is precisely the number of faces in the interior of the cycle.

3.3 Deterministic Isolation

The non-vanishing circulations immediately give us the isolation for the perfect matchings.

Lemma 3.2 Isolation *If all the cycles in a bipartite graph have non-zero circulations, then the minimum weight perfect matching in it is unique.*

Proof If not, then we have two minimum weight perfect matchings M_1 and M_2 which will contain some alternating cycles, and each such cycle is of even length. Consider any one such cycle. Since the circulation of an even cycle is nonzero either the part of it which is in M_1 is lighter or the part of it which is in M_2 is lighter, in either case, we can form another perfect matching which is lighter than the minimum weight perfect matching, which is a contradiction. □

Thus we have a deterministic way of isolating a perfect matching in bipartite planar graphs, and it is easy to check that the procedure of assigning the weights to the edges works in deterministic Logspace.

3.4 Extracting the Unique Matching

Once we have isolated a perfect matching, one can extract it out easily as follows:

- **Step 1:** Construct an $n \times n$ matrix M associated with a planar graph on n vertices as follows: Find a *Pfaffian orientation* [10] of the planar graph and put the (i, j)th entry of the matrix M to be $x^{w_{(i,j)}}$ where x is a variable and $w_{(i,j)}$ is the weight of the edge (i, j) which is directed from i to j in the Pfaffian orientation. If the edge is directed from j to i then put $-x^{w_{(i,j)}}$ as (i, j)th entry of the matrix.
- **Step 2:** If t is the weight of the minimum weight perfect matching, then the coefficient of x^{2t} in determinant of M will be the number of perfect matchings of weight t. Now, as shown in [16, 20] this coefficient can be written as a determinant of another matrix.
- **Step 3:** Now start querying from $i = -n^2$ to $+n^2$ whether the coefficient of x^{2i} is zero or not and the first time you find that it is nonzero; stop. The first nonzero value will occur when $i = t$ and it will be 1 since the minimum weight perfect matching is unique. Hence, during the process, every time we have a promise that if the determinant is nonzero, it is 1. This procedure gives the weight of the minimum weight perfect matching.
- **Step 4:** Now once we know the procedure to find the weight of the minimum weight perfect matching, then one can find out which edges are in the matching by simply deleting the edge and again finding the weight of the minimum weight perfect matching in the remaining graph. If the edge is in the isolated minimum weight perfect matching then after its deletion the weight of the new minimum weight perfect matching will increase. Otherwise we can conclude that the edge is not in the isolated minimum weight perfect matching. Note that the isolation holds even after deleting an edge from the graph.

Theorem 3.3 *Bipartite Planar Perfect Matching is in* SPL.

Proof The Logspace procedure in Lemma 3.1 assigns the small weights to the edges of the graph so that the minimum weight perfect matching is unique and the above procedure extracts it out in $L^{\mathsf{SPL}} = \mathsf{SPL}$. □

We obtain the following corollaries.

Corollary 3.4 UPM *in bipartite planar graphs is in* SPL.

Proof (Earlier bound for this was $\oplus L$ by [6].) To check whether the graph has unique perfect matching, one can construct one perfect matching and check that after removing any edge of it there is no other perfect matching. □

Corollary 3.5 *Minimum weight perfect matching in planar graphs with polynomially bounded weights is computable in* SPL.

Proof One can first scale the polynomially bounded weights by some large multiplicative factor, say n^4 and then perturb them using the weighting scheme described above so that some minimum weight matching with original weights remains minimum weight matching with new weights and is unique. Then extraction can be done in SPL. □

Corollary 3.6 *Minimum weight spanning tree in planar graphs is computable in* SPL *if the weights are polynomially bounded.* (*The same is true for directed rooted spanning trees* (*arborescences*) *in planar graphs due to Generalized Temperley's bijection shown in* [11].)

Proof This follows from Temperley's bijection and Generalized Temperley's bijection in [11]. □

Note that, deciding whether a planar graph has a spanning tree is known to be in Logspace whereas deciding whether a directed planar graph has an arborescence rooted at a particular node is inside UL as it reduces to checking directed reachability from the root to all other nodes. The minimum weight arborescence question is interesting in the context of matching as structures similar to *blossoms* come as a natural obstacle while trying to parallelize. It is interesting to note that the Generalized Temperley bijection helps us to avoid these *blossoms* in case of the minimum weight spanning arborescence for directed planar graphs.

4 Complexity of Further Restrictions and Very Specialized Cases

In this section we consider several restricted class of graphs and study the complexity of perfect matching questions there. We study decision version, search version as well as UPM (is there a unique perfect matching?) version of the perfect matching questions. The motivation is to identify natural subclasses of graphs for which this complexity is close to Logspace and to get an intuition about whether the complexity of (bipartite) planar matching is likely to be significantly lower than our upper bound of SPL. For instance, is bipartite planar perfect matching in Logspace? Table 1 summarizes all the results that we have.

Restricting the family of planar graphs, yields better upper bounds for Matching questions. Notably, we prove that:

Corollary 4.1 (*of Theorem* 3.3) *Perfect matching in outerplanar graphs is in* SPL.

Table 1 Very specialized restrictions

Restriction	Decision	Search	UPM
General Planar	NC	P	NC
New	NC	P	NC
Biparite Planar	NC	NC	$\oplus L$
New	SPL	SPL	SPL
Outerplanar	NC	P	NC
New	**LogCFL** $\cap$ SPL	SPL	**L**
Series Parallel Graphs	NC	P	NC
New	**LogCFL**	P	$\mathbf{LogCFL}^{\oplus L}$
$\sqrt{\log n}$ Width Layered Graphs	P	P	P
New	**NL**	P	$\mathbf{NL}^{\oplus L}$

Proof If we have an outerplanar (1-page) graph on n vertices with vertices labeled from 1 to n along the spine, then observe that the edges between two odd labeled vertices cannot be part of any perfect matching. This is because the number of vertices below that edge is odd. Similarly edges between two even labeled vertices cannot participate in any perfect matching. Hence, by removing such edges we can make the graph bipartite and then we can apply the previous theorem. □

Definition 4.2 Given a graph G the Laplacian matrix of G denoted by $L(G)$ is defined as $D - A$ where D is the diagonal matrix of degrees of vertices in G and A is the adjacency matrix of G.

We use the lemma below in order to prove the theorem that follows it.

Lemma 4.3 *The parity of the number of perfect matchings in an outerplanar graph can be computed in Logspace.*

Notice that parity refers to the function that maps even integers to 0 and odd integers to 1.

Proof It is not hard to observe that the parity of the determinant of the adjacency matrix of a graph is the same as the parity of number of perfect matchings in it. Finding the parity of the adjacency matrix of an outerplanar graph can be reduced to finding the parity of the number of spanning trees in an auxiliary planar graph which is constructed by adding a new vertex and connecting it to all the odd degree vertices of the original graph. The new graph has all the vertices of even degree. Hence the adjacency matrix of the new graph is the same as its Laplacian modulo 2. Now the minor obtained by removing the row and the column corresponding to the new vertex, is precisely the adjacency matrix of the original outerplanar graph modulo 2. Also the determinant (mod 2) of this minor is precisely the parity of the spanning trees in new graph. As shown in [4], the parity of spanning trees modulo 2 in planar graphs can be computed in Logspace. Hence, the parity of the determinant of the adjacency matrix of an outerplanar graph can be obtained in Logspace which in turn gives the parity of the number of perfect matchings in it. □

Theorem 4.4 UPM *in outerplanar graphs is in Logspace.*

Proof If the perfect matching in an outerplanar graph is unique, one can obtain one perfect matching in Logspace. For every edge, one just needs to compute the parity (whether the number is odd or even) of the number of perfect matchings in the graph with that particular edge removed. If this parity is 0 (number is even) then do not include this edge in the perfect matching, otherwise do. Now we just need to verify that the perfect matching thus constructed is unique. As seen in Corollary 4.1 we can assume that the outerplanar graph is bipartite. Given a bipartite graph and a perfect matching one can construct an auxiliary directed graph as follows: if u to w is a matching edge and v to w is a non-matching edge, then draw a directed edge from u to v. It turns out that the auxiliary graph has a directed cycle if and only if the perfect matching in the original graph was not unique. It is also easy to see that if you start with a bipartite outerplanar graph and a perfect matching in it then the corresponding auxiliary directed graph will be outerplanar. Finally, since the reachability in directed outerplanar graphs is in Logspace [1], we have that UPM in outerplanar graphs is in Logspace. □

Outerplanar graphs are known to have constant tree-width. It is easy to observe the following theorem about constant-tree-width graphs.

Lemma 4.5 *Deciding whether a constant-tree-width graph has a perfect matching is in LogCFL.*

Proof Firstly, one can obtain a tree decomposition of constant width of such graphs in LogCFL, see [9]. Once we have a tree decomposition, we make an Euler traversal on the tree everytime guessing the edges of the matching within the constant size bag corresponding the node of the tree. We store our guess on a stack and verify it while we return to the node through the Euler traversal. □

The construction version, though, for constant-tree-width graphs is not known to be in NC. However, we can note that UPM version is parallelizable.

Corollary 4.6 *UPM in constant-tree-width graphs is in LogCFL$^{\oplus L}$.*

Proof First by finding out the parity of the determinant of the adjacency matrix of the graph one can obtain a perfect matching. Then one can verify the uniqueness by checking that after removing any matching edge, the graph does not have any perfect matching. Note that LogCFL is closed under complement. □

Corollary 4.7 *Decision version of perfect matching question in series parallel graphs is in LogCFL.*

Proof Series parallel graphs have tree width 2. □

Lemma 4.8 *Decision version of perfect matching in a layered graph with each layer of $\sqrt{\log n}$ size and the edges present only between two consecutive layers*; *is in* NL.

Proof One can guess a perfect matching layer by layer as a bit vector of logarithmic length one for each edge in the layer indicating the status picked or not picked in matching and verify it along the way. □

Corollary 4.9 *UPM question in such graphs is in* $\mathsf{NL}^{\oplus L}$.

Proof Once one gets a perfect matching in $\oplus L$, we can check the uniqueness of the perfect matching by checking for every edge whether there is a perfect matching in the graph containing that edge and there is no perfect matching after removing that edge. Since NL is closed under complement the effective complexity is $\mathsf{NL}^{\oplus L}$. □

Corollary 4.10 *Perfect-Matching-Decision for subgraphs of $\sqrt{\log n}$ width grid with possibly diagonal edge in each square is in* NL. *Moreover the corresponding UPM question is in* $\mathsf{NL}^{\oplus L}$.

Proof Such a graph is a layered graph with each layer of $\sqrt{\log n}$ width and the edges are only between two consecutive layers, hence by the previous theorem the result follows. □

5 Possibility of Generalization

We saw in Sect. 3.3 that a perfect matching in bipartite planar graphs can be isolated by assigning small weights to the edges. In this section we discuss the possibility of generalizing this result in two orthogonal directions. For non-bipartite planar graphs and for bipartite but non-planar graphs. The motivation is to isolate a perfect matching in a graph by having non-zero circulation on it.

5.1 Is Non-Vanishing Circulation Possible for Arbitrary Graphs?

It is not clear whether the method of non-vanishing circulations can be generalized for arbitrary graphs. Even the existence of a weighting scheme that can achieve non-vanishing circulations using small weights is not known. Asking for non-vanishing circulations is stronger than asking for isolating a perfect matching. It is still open whether one can isolate a perfect matching in non-bipartite planar graphs.

Open Problem 5.1 Does there exist a weighting scheme with small weights which achieves non-vanishing circulations for all even cycles in a planar graph?

Open Problem 5.2 Does there exist a weighting scheme with small weights which achieves non-vanishing circulations for all cycles in bipartite graphs?

5.2 Non-Bipartite Planar Matching

Though non-bipartiteness is an issue, we can get rid of it to a certain extent. Although as expected, not completely.

Lemma 5.3 *Perfect Matching problem in general planar graphs reduces to that of almost grid graphs.*

Now it suffices to get a non-vanishing circulations in almost grid graphs to solve planar matching question. Unfortunately we don't yet know any way of achieving this though we have some observations which might be helpful.

Lemma 5.4 *One can have non-zero circulations for all the even cycles in the graph in Fig.* 4. (*The graph is simply one row of the grid with diagonals.*)

Proof Observe that any even cycle in such a graph will either fall in the grid or will fall in the grid formed by horizontal edge together with diagonal edges. Now, its easy to assign the weights as shown in Fig. 4 so that all the horizontal edges get weight 0 while vertical and diagonal edges get weights so that any cycle in vertical or diagonal grid has non-vanishing circulation. □

Lemma 5.5 *One can have a non-zero circulation for an almost grid graph which has the bottom row as the only non-bipartite row.*

Proof Use two co-ordinates for the weights. In the first co-ordinate, the grid edges except for the last row will get the same weighting as the case of grid graphs except that the vertical edges become horizontal and the horizontal edges become vertical. Edges of the last row of the grid get weight zero in the first co-ordinate. In the second co-ordinate, the last row gets the weights as given in previous lemma. Grid edges in the second co-ordinate get weight according to grid weights with vertical edges becoming horizontal. Now, for any cycle that does not use the diagonal edges, the circulation is non-vanishing because the cycle will totally fall into the grid and the second co-ordinate will be non-vanishing. Any even cycle that stays only within last row will also get nonzero circulation as seen in previous lemma. Any cycle that passes through higher rows can be decomposed into the arcs, the portion of the cycle that fall strictly above the bottom row of edges (it might contain the edges of the penultimate row). Now for each such arc, the alternate sum along the arc will compute in absolute value the number of blocks of the grid below it if we look at the weights in the first co-ordinate. Now, the topmost arc has to be unique because consider the leftmost

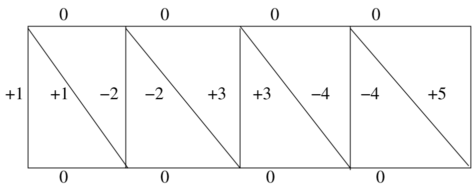

Fig. 4 Non-vanishing circulation in a non-bipartite graph

point where the cycle leaves the bottommost row of vertices and then the first time it comes back to the bottommost row, the rest of the cycle has to fall inside this arc. Moreover it is easy to see that there can be only two layers of the arcs there is no way to accommodate the third layer of arch inside an arc in the second layer. Thus, no matter what signs the alternating sum along these arcs acquire, the total sum is not going to cancel out in the first co-ordinate because the topmost arc is going to survive. □

In summary, non-bipartiteness seems to be an issue which is difficult to get around. Hence, we keep the bipartiteness and next we explore the possibility of generalizing our result for non-planar graphs.

5.3 Bipartite Non-Planar Matching

Instead of looking at two dimensional grid we now consider three dimensional grids. The matching problem remains as hard as that for general bipartite graphs in this restriction as well.

Lemma 5.6 *One can embed any bipartite graph in a three dimensional grid while preserving matchings.*

Proof Firstly, one can make the degree of the graph bounded by 3. Then one can use the third dimension to make the space for crossings in the graph. □

Open Problem 5.7 Is the perfect matching problem for subgraphs of a three dimensional grid of height 2 in NC? Is it possible to generalize our method from grid graphs to 3-dimensional grid graphs of height 2?

The deterministic isolation of perfect matching is possible through non-vanishing circulations as seen in Sect. 3.3.

5.4 Bit Complexity of the Method of Non-Vanishing Circulations

It is not-known whether $\log n$ bits are necessary to isolate a perfect matching in planar graphs or even in general graphs. Our method for obtaining non-vanishing circulations in grid graphs relied crucially on the following observation: It suffices to assign the weights so that:

(1) Circulation of every block is non-zero.
(2) Circulations of adjacent blocks are opposite in sign.

We show that this method would require $\Omega(\log n)$ bits for some edge.

Theorem 5.8 *Any weighting scheme for an $n \times n$ grid that achieves nonzero circulation for every block such that circulations of adjacent blocks are opposite in signs, has to use $\Omega(\log n)$ bit weights for some edge of the grid.*

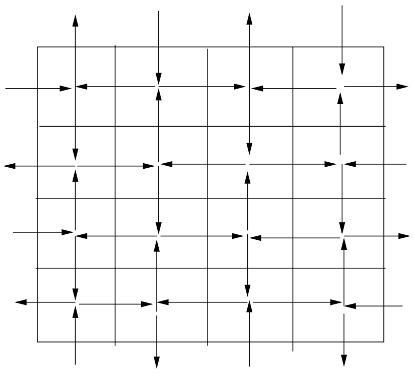
Fig. 5 An orientation of dual grid

Proof Consider the dual graph. Construct an orientation of the dual graph as follows: for the dual vertex corresponding to the bottom-left corner block of the grid, vertical dual edges go outwards whereas the horizontal dual edges go inwards. For its adjacent dual vertex, vertical dual edges go inwards whereas the horizontal dual edges go outwards. See Fig. 5. Thus we get a directed graph with the weights on its directed edges inherited from the weights of the original graph.

Flux out of a vertex in the directed graph is the sum of the weights of the outgoing edges minus the sum of the weights of the incoming edges at that vertex. Now, flux out of every vertex in the graph is positive because the orientations of the edges are chosen such that the flux out of a vertex is precisely sign times circulation of the corresponding face in the primal graph. So, total of $\Omega(n^2)$ flux has to pass through the perimeter of the $n \times n$ grid which is $O(n)$. Thus, some edge on the perimeter must get a weight of $\Omega(n)$. □

5.5 Isolation for Matching and Directed Paths

In this section we will consider two methods used for deterministic isolation. In [5] the method of *non-vanishing directed cycle sums* is used to isolate directed paths in planar graphs. In this paper, we use the method of *non-vanishing circulations* to isolate a perfect matching in bipartite planar graphs.

Definition 5.9 (Non-Vanishing Directed Cycle Sums) Given an undirected graph, consider each undirected edge as two edges directed in each direction. *Non-vanishing directed cycle sums* is a skew symmetric ($w(i,j) = -w(j,i)$) weighting of the directed edges using logarithmic bit long weights such that the sum of the weights of the directed edges along any directed cycle is not zero.

We show that in any class of graphs that is closed under subdivision of edges, non-vanishing circulations lead to non-vanishing cycle sums.

Theorem 5.10 *For a class of graphs which is closed under subdivision of edges, suppose we can achieve non-vanishing circulation for all bipartite graphs in the class, then we can achieve non-vanishing cycle sum for all graphs in the same class.*

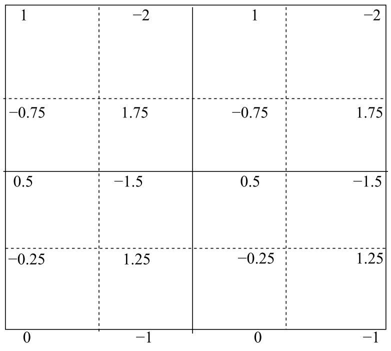

Fig. 6 A non-vanishing circulation yielding exactly weighting from [5] for isolating directed paths in grid graphs

Proof Given a graph, construct a bipartite graph by replacing each directed edge by a path of length 2. Suppose that the edge e gets replaced by two edges e_1 and e_2. Now, obtain a non-vanishing circulation for this bipartite graph. Suppose that the edge e_1 gets weight a and e_2 gets weight b. Assign the weight of e in one direction to be $a - b$ and in other direction $b - a$. Now, directed cycle sum in the original graph will correspond exactly to the circulations in the new graph and thus are non-vanishing. □

Corollary 5.11 *Non-vanishing directed cycle sums for planar graphs can be obtained by a Logspace procedure.*

Proof Non-vanishing circulations for bipartite planar graphs will give non-vanishing directed cycle sum for arbitrary planar graphs. □

Non-vanishing circulations in Fig. 6 give exactly the weighting used by [5] for isolating directed paths in grid graphs. To see this, four blocks constitute one block of the grid as the procedure above splits every edge into two. All vertical edges have weight zero. Every horizontal edge consists of two horizontal edges of weight say a and b from left to right. The directed edge going from left to right gets weight $a - b$ whereas the opposite edge gets weight $b - a$.

An alternate way of obtaining non-vanishing circulations is to orient the edges of the grid graph in a nice way and then use the weighting scheme used in [5] for directed grid graphs on the oriented graph. We call this a projection from directed to undirected graphs, i.e., choice a direction for every undirected edge. Figure 7 gives such a projection which gives non-vanishing circulations. To see that indeed this projection gives non-vanishing circulations one has to essentially use the block decomposition of circulations. In general graphs it is not clear whether non-vanishing directed cycle sums give rise to non-vanishing circulations, i.e, whether such projection is possible. Reachability reduces to bipartite matching [12] but the other way is not known, even when restricted to planar graphs. The projection from weighting scheme of [5] to obtain non-vanishing circulations for grid, however, implies a lower bound of $\Omega(\log n)$ on the size of the weights obtained by following the method of [5], namely to assign weights so as to make anti-clockwise cycle sum for every block to be positive. Figure 7 describes the projection from [5] to obtain non-vanishing circulations.

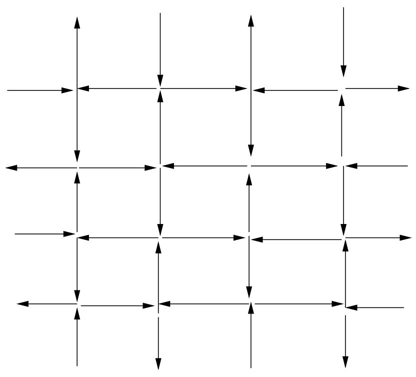
Fig. 7 A projection from weights from [5] yielding non-vanishing circulations for grid

5.6 Other Variations

We know that reachability in directed planar graphs reduces to bipartite planar matching. Note that while the isolation step in our algorithm works in Logspace, placing the bipartite planar perfect matching problem in Logspace might be too strong a result to expect. This seems like a possible but not very promising approach to place planar reachability in Logspace.

However, we also know that reachability in layered grid graphs reduces to the UPM question in the same [6]. A Logspace upper bound on Bipartite Planar UPM might very well be achievable and Theorem 4.4 yields some (weak) substantiation for that. Thus a positive resolution of the following question would place layered grid graph reachability in Logspace.

Open Problem 5.12 Is bipartite planar UPM in L?

We saw how to isolate a perfect matching in bipartite planar graph. The isolation holds for maximum matching in bipartite planar graphs. However, we do not know how to extract out the maximum weight perfect matching in NC.

Open Problem 5.13 Is it possible to extract out the isolated maximum matching in NC for bipartite planar graphs?

Finally, the question of constructing a perfect matching in planar graphs in NC still remains open.

Acknowledgements We thank Meena Mahajan for useful discussion and comments on the preprint. We would also like to thank the anonymous referees for their help in improving the presentation of the paper.

References

1. Allender, E., Barrington, D.A.M., Chakraborty, T., Datta, S., Roy, S.: Grid graph reachability problems. In: IEEE Conference on Computational Complexity, pp. 299–313 (2006)
2. Allender, E., Datta, S., Roy, S.: The directed planar reachability problem. In: FSTTCS, pp. 238–249 (2005)

3. Allender, E., Reinhardt, K., Zhou, S.: Isolation, matching, and counting: uniform and nonuniform upper bounds. J. Comput. Syst. Sci. **59**(2), 164–181 (1999)
4. Braverman, M., Kulkarni, R., Roy, S.: Parity problems in planar graphs. In: IEEE Conference on Computational Complexity, pp. 222–235 (2007)
5. Bourke, C., Tewari, R., Vinodchandran, N.V.: Directed planar reachability is in unambiguous log-space. In: IEEE Conference on Computational Complexity, pp. 217–221 (2007)
6. Datta, S., Kulkarni, R., Limaye, N., Mahajan, M.: Planarity, determinants, permanents, and (unique) matchings. In: CSR, pp. 115–126 (2007)
7. Diestel, R.: Graph Theory. Springer, Berlin (2005)
8. Grigoriev, D., Karpinski, M.: The matching problem for bipartite graphs with polynomially bounded permanent is in NC. In: Proceedings of 28th IEEE Conference on Foundations of Computer Science, pp. 166–172. IEEE Computer Society Press, Los Alamitos (1987)
9. Gottlob, G., Leone, N., Scarcello, F.: Theor. Comput. Sci. Arch. **270**(1–2), 761–777 (2002)
10. Kasteleyn, P.W.: Graph theory and crystal physics. In: Harary, F. (ed.) Graph Theory and Theoretical Physics, pp. 43–110. Academic Press, New York (1967)
11. Kenyon, R.W., Propp, J.G., Wilson, D.B.: Trees matchings. Electron. J. Comb. **7**(1)
12. Karp, R., Upfal, E., Wigderson, A.: Constructing a perfect matching is in random NC. Combinatorica **6**, 35–48 (1986)
13. Kozen, D., Vazirani, U.V., Vazirani, V.V.: NC algorithms for comparability graphs, interval graphs, and testing for unique perfect matching. In: FSTTCS, pp. 496–503 (1985)
14. Lovasz, L., Plummer, M.: Matching Theory. North-Holland, Amsterdam (1986)
15. Miller, G., Naor, J.: Flow in planar graphs with multiple sources and sinks. SIAM J. Comput. **24**, 1002–1017 (1995)
16. Mahajan, M., Vinay, V.: A combinatorial algorithm for the determinant. In: SODA, pp. 730–738 (1997)
17. Mahajan, M., Varadarajan, K.: A new NC algorithm to find a perfect matching in planar and bounded genus graphs. In: Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing (STOC), pp. 351–357 (2000)
18. Mulmuley, K., Vazirani, U., Vazirani, V.: Matching is as easy as matrix inversion. Combinatorica **7**(1), 105–131 (1987)
19. Vazirani, V.: NC algorithms for computing the number of perfect matchings in $K_{3,3}$-free graphs and related problems. In: SWAT, pp. 233–242 (1988)
20. Vollmer, H.: Introduction to Circuit Complexity—A Uniform Approach; Texts in Theoretical Computer Science. An EATCS Series. Springer, Berlin (1999)