Deterministically Isolating a Perfect Matching in Bipartite Planar Graphs

Samir Datta, Raghav Kulkarni, Sambuddha Roy

Symposium on Theoretical Aspects of Computer Science 2008 (Bordeaux), pp. 229-240
www.stacs-conf.org

DETERMINISTICALLY ISOLATING A PERFECT MATCHING IN BIPARTITE PLANAR GRAPHS

SAMIR DATTA [1], RAGHAV KULKARNI [2], AND SAMBUDDHA ROY [3]

[1] Chennai Mathematical Institute, Chennai, India.
E-mail address: sdatta@cmi.ac.in

[2] University of Chicago, Chicago, USA.
E-mail address: raghav@cs.uchicago.edu

[3] IBM Research Laboratory, New Delhi, India.
E-mail address: sambuddha@in.ibm.com

Abstract. We present a deterministic way of assigning small (log bit) weights to the edges of a bipartite planar graph so that the minimum weight perfect matching becomes unique. The *isolation lemma* as described in [MVV87] achieves the same for general graphs using a randomized weighting scheme, whereas we can do it deterministically when restricted to bipartite planar graphs. As a consequence, we reduce both decision and construction versions of the matching problem 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 bipartite planar graphs. This improves the earlier known bounds of non-uniform SPL by [ARZ99] and NC^2 by [MN95, MV00]. 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. Our techniques are elementary and simple.

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 applications even outside the field. 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.

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

SYMPOSIUM ON THEORETICAL ASPECTS OF COMPUTER SCIENCE

1.1. The Matching Problem

Definition 1.1. A matching in an undirected graph is a collection of edges which have no endpoint in common.

Such a collection of edges is called "independent". See [LP86] for an excellent survey on matchings.

The 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 collection of independent edges which cover all the vertices.

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 [KVV85] (i.e. it lies in the complexity class NC; see any standard complexity text for a formal definition, say [V99]). Intuitively, NC is a complexity class consisting of the 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, [KUW86, MVV87]. Several restrictions of the matching problem are known to be in NC, for example, bipartite planar graphs [MN95, MV00], graphs with polynomially bounded number of perfect matchings [GK87] etc. Whether the search version reduces to the decision version has also not been answered yet.

1.2. Randomized Isolation Lemma

Lemma 1.4. [MVV87] *One can randomly assign polynomially bounded weights to the edges of a graph so that with high probability the minimum weight perfect matching becomes unique.*

Using the isolation lemma, [MVV87] obtained a simple *Randomized* NC algorithm for finding a perfect matching in arbitrary graphs.

1.3. Matching in SPL/poly

Allender et al [ARZ99] 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 polynomial bit advice.

Definition 1.5. SPL is a promise class that is characterized by the problem of checking whether a a matrix is singular under the promise that its determinant is either 0 or 1. The corresponding non-uniform class SPL/poly is SPL with a polynomial bit advice.

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

Definition 1.6. A language is said to be in SPL/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 membership of any input of size n can be decided in SPL.

Theorem 1.7. [ARZ99] *Matching is in* SPL*/poly.*

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 ([K67, V88]) whereas constructing one perfect matching is not yet known to be parallelizable. However, for bipartite planar graphs, people have found NC algorithms [MN95, MV00].

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 [BTV07] building upon a technique from [ADR05] 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.

Motivated by their result we explore the possibility of such an isolation for perfect matchings in planar graphs. 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 cycle result from the super-imposition 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.

2. Preliminaries

Here we describe the technical tools that we need in the rest of the paper. Refer to any standard text (e.g. [V99]) for definitions of the complexity classes $\oplus$L, $\oplus_p$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. [D05]).

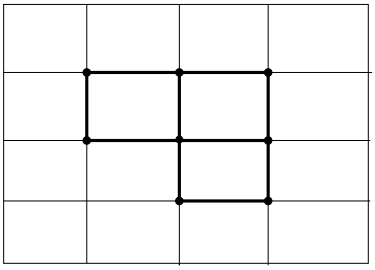

Figure 1: A Grid Graph

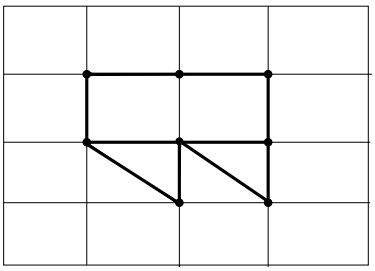

Figure 2: An Almost Grid Graph

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 Figure 1 for an example. We call each unit square of the grid a *block*.

Definition 2.2. 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 Figure 2.

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

Definition 2.3. (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 i^{th} row and j^{th} column (adjacent blocks get opposite signs).

Definition 2.4. Given a weighted grid graph G, the *circulation* of a block B(denoted $circ(B)$) in G is the signed sum of weights of the edges of it: $circ(B) = \sum_{e \in B} sign(e) weight(e)$.

Definition 2.5. Given a weighted grid graph G and a simple 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 $circ(C) = \sum_{i=0}^{2k-1} (-1)^i weight(e_i)$.

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

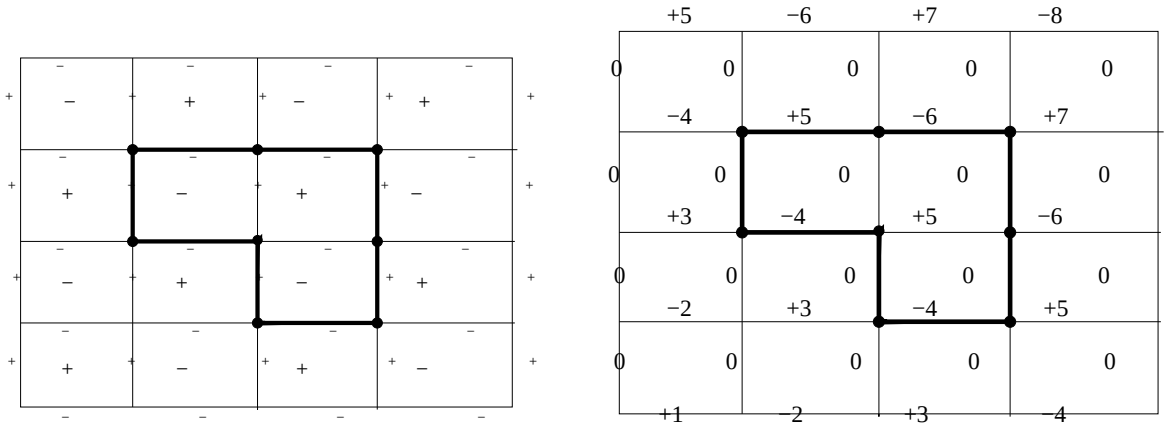

Figure 3: Signs and Weights of the blocks and the edges of a grid

Lemma 2.6. Block Decomposition of Circulations: *The absolute value of the circulation of a simple 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.*
$|circ(C)| = \sum_{B \in interior(C)} sign(B) circ(B)$

Proof. Consider the summation on the right hand side. The weight of any edge in the *interior* of C will get cancelled 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 2.7. *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 Figure 3. In the other case, when adjacent boundary edges do not lie on adjacent blocks, they lie on two 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. See Figure 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 + $circ(C)$ or - $circ(C)$. ■

■

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

Lemma 2.8. 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 [KPW00] 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. [ABCDR06, BTV07]. 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 2.9. *One can convert any bipartite planar graph into a grid graph such that the perfect matchings remain in one-to-one correspondence.*

Proof. This is described in [DKLM07]. It follows closely the procedure for embedding a planar graph into a grid, described by [ABCDR06]. ∎

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

Lemma 2.10. *Any planar graph, not necessarily bipartite, can be converted 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. ∎

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 [MN95, MV00]. 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.

Lemma 3.1. *One can assign, in Logspace, small (logarithmic bit) 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 the Figure 3.)*

Proof. We assign all vertical edges weight 0 and horizontal edge from grid point (i, j) to $(i+1, j)$ is assigned a weight of $(-1)^{i+j}(i+j+1)$ as shown in figure 3. Thus the circulation of the block with diagonally opposite vertices (i, j) and $(i+1, j+1)$ is $\sum_e sign(e)weight(e)$ $= (-1)(-1)^{i+j}(i+j+1) + (+1)0 + (-1)(-1)^{i+j+1}(i+j+2) + (+1)0 = (-1)^{i+j}$

Thus, 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 2.6), 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

One can think of the procedure of assigning the weights to the edges of bipartite planar graph without having to convert it to a grid graph. The procedure is as follows:

- **Step 1:** *Make the graph Eulerian (every vertex has an even degree):* add spurious edges to it without disturbing the bipartiteness.
  - *Step 1.1:* Perform an Euler traversal on a spanning tree in the dual graph.
  - *Step 1.2:* While performing the traversal, make sure that when you leave the face, all the vertices in the face, except for the end points of the edge through which we go to the next face, are of even degree. To guarantee this we can do the following:
    - *Step 1.2 a) :* 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 1.2 b) :* If the next vertex is also of odd degree then go to its next vertex. If the next vertex is of even degree then we have pushed the oddness one step further.
    - *Step 1.2 c) :* Continue the same procedure till we remove all the oddness or push it to v.
    - *Step 1.2 d) :* 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.
- **Step 2:***Fix the signs:* After Step 1, the graph has become Eulerian, and hence the dual graph is bipartite.
  - *Step 2 a) 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 2 b) 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. Now since both the original graph and its dual are bipartite , the auxiliary graph will also be bipartite - 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:* 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 (logarithmic bit).

Block Decomposition of Circulations: Again, the circulation of a cycle will decompose into signed sum of circulations of the faces in the interior of it and since the sign and the circulation for any face is the same, we will have non-vanishing circulations in the graph. The details are analogous to the case of a grid. We leave the details to the reader.

3.3. Deterministic Isolation

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

Lemma 3.2. *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* ([K67]) 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 [MV97, V99] 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 non-zero, 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 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. 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* [KPW00]*.)*

Proof. This follows from Temperley's bijection. ∎

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

Corollary 3.7. *(of Theorem 3.3) Perfect Matching in outerplanar graphs is in* SPL.

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

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

Lemma 3.8. *The parity of the number of perfect matchings in an outerplanar graph can be computed in Logspace.*

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 [BKR07], 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 3.9. 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 of the number of perfect matchings in the graph with that particular edge removed. If this parity is 1 then don't 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 3.7 we can assume that the outerplanar graph is bipartite. Now, if we consider an auxiliary directed graph obtained from this outerplanar graph by putting a directed edge starting from a vertex and ending in another vertex after following a matching edge starting at the vertex and then a non-matching edge from there, then, this auxiliary graph will have a directed cycle if and only if the matching we start with is not unique. It is possible to show that the auxiliary graph is outerplanar. Finally, since the reachability in directed outerplanar graphs is in Logspace ([ABCDR06]), we have that UPM in outerplanar graphs is in Logspace. ∎

4. Discussion

We saw in Section 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.

4.1. Non-bipartite Planar Matching

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

Lemma 4.1. *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 4.2. *One can have non-zero circulations for all the even cycles in the graph in the Figure 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 the Figure 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. ∎

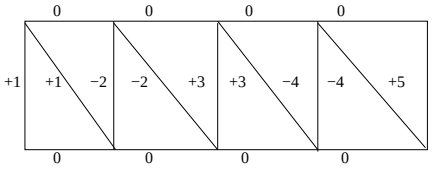

Figure 4: Non-vanishing circulation in a non-bipartite graph

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.

4.2. 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 4.3. *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 4.4. Is the perfect matching problem for subgraphs of a three dimensional grid of height 2 (constant height in general) in NC ?

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

Open Problem 4.5. Is small (log bit) weight non-vanishing circulation possible in every bipartite graph?

4.3. Other Variations

We know that the reachability in directed planar graphs reduces to bipartite planar matching while the reachability in layered grid graphs reduces to the UPM question in the same [DKLM07]. Note that the isolation step in our algorithm works in Logspace. Solving the perfect matching question in bipartite planar graphs in Logspace might be too strong to expect but at least one can ask the question about UPM which would put layered grid graph reachability in Logspace or giving an orthogonal bound for the same.

Open Problem 4.6. 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 4.7. Is it possible to extract out the isolated maximum matching in NC even for bipartite planar graphs?

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

5. Acknowledgements

We thank Meena Mahajan for useful discussion and comments on the preprint.

References

[ABCDR06] Eric Allender, David A. Mix Barrington, Tanmoy Chakraborty, Samir Datta, Sambuddha Roy: Grid Graph Reachability Problems. IEEE Conference on Computational Complexity 2006: 299-313

[ADR05] Eric Allender, Samir Datta, Sambuddha Roy: The Directed Planar Reachability Problem. FSTTCS 2005: 238-249

[ARZ99] Eric Allender, Klaus Reinhardt, Shiyu Zhou: Isolation, Matching, and Counting: Uniform and Nonuniform Upper Bounds. J. Comput. Syst. Sci. 59(2): 164-181 (1999)

[BKR07] Mark Braverman, Raghav Kulkarni, Sambuddha Roy: Parity Problems in Planar Graphs. IEEE Conference on Computational Complexity 2007: 222-235

[BTV07] Chris Bourke, Raghunath Tewari, N. V. Vinodchandran: Directed Planar Reachability is in Unambiguous Log-Space. IEEE Conference on Computational Complexity 2007: 217-221

[DKLM07] Samir Datta, Raghav Kulkarni, Nutan Limaye, Meena Mahajan: Planarity, Determinants, Permanents, and (Unique) Matchings. CSR 2007: 115-126

[D05] Reinhard Diestel. Graph Theory. *Springer*, 2005

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

[K67] P. W. Kasteleyn. Graph theory and crystal physics. In F. Harary, editor, *Graph Theory and Theoretical Physics*, page 43-110, Academic Press, 1967.

[KPW00] Richard W. Kenyon, James G. Propp, David B. Wilson, Trees and Matchings, Electronic Journal of Combinatorics, 7(1)

[KUW86] Richard Karp, Eli Upfal, Avi Wigderson. Constructing a perfect matching is in random NC. *Combinatorica, 6:35-48*, 1986.

[KVV85] Dexter Kozen, Umesh V. Vazirani, Vijay V. Vazirani. NC Algorithms for Comparability Graphs, Interval Graphs, and Testing for Unique Perfect Matching. FSTTCS 1985: 496-503

[LP86] L. Lovasz, M. Plummer, Matching Theory, North-Holland, 1986.

[MN95] Gary Miller and Joseph Naor. Flow in planar graphs with multiple sources and sinks. *SIAM Journal of Computing*, 24:1002-1017, 1995.

[MV97] Meena Mahajan, V. Vinay: A Combinatorial Algorithm for the Determinant. SODA 1997: 730-738

[MV00] Meena Mahajan, Kasturi Varadarajan. 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)*, pages 351-357, 2000.

[MVV87] Ketan Mulmuley, Umesh Vazirani, Vijay Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1): 105-131, 1987.

[V88] Vijay Vazirani. NC Algorithms for Computing the Number of Perfect Matchings in $K_{3,3}$–free Graphs and Related Problems. SWAT 1988: 233-242

[V99] Heribert Vollmer, Introduction to Circuit Complexity - A Uniform Approach; Texts in Theoretical Computer Science. An EATCS Series. Springer Verlag, 1999.