

# MATCHING IN PLANAR GRAPHS

*A Thesis Submitted*  
*in Partial Fulfillment of the Requirements*  
*for the Degree of*  
**Master of Technology**

*by*  
**Arpita Raghavendra Korwar**  
**Y7111018**



*to the*  
**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**  
**INDIAN INSTITUTE OF TECHNOLOGY KANPUR**  
June, 2009



# CERTIFICATE

This is to certify that the work contained in the thesis entitled “*Matching in Planar Graphs*” by *Arpita Raghavendra Korwar* has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

June, 2009

Piyush P. Kurur

Department of Computer Science and Engineering

Indian Institute of Technology Kanpur

Kanpur 208016



# Abstract

Matching is one of the most well-studied combinatorial problems. Deterministic polynomial time, as well as randomized NC algorithms that find a perfect matching in a graph are known. But we don't know of any efficient parallel algorithm that would solve this problem. There exist NC algorithms that can solve the perfect matching problem in bipartite planar graphs. In this work, we try to find an NC algorithm that searches for a perfect matching in planar graphs.

We try to derandomize the algorithm based on the Isolation lemma [MVV87] and find a weight assignment to the edges such that a unique perfect matching with the minimum weight exists. We define the even cycle subspace and present an NC algorithm that finds its basis.

We also provide a simpler version of the algorithm presented by Kulkarni, et al. [DKR08], which finds the perfect matching in planar bipartite graphs.

The even cycle subspace is a generalization of the cycle space in bipartite graphs. The cycle space in bipartite graphs can be used to find the perfect matching in bipartite planar graphs. Hence, though we have not been able to solve the problem we hope that the ideas presented here would be useful in finding an NC algorithm that searches a perfect matching in planar graphs.



# Acknowledgements

*I am highly grateful to **Prof. Piyush P. Kurur** for his encouragement and support. I consider myself fortunate to have had a patient and enthusiastic Thesis guide like him. He even helped me pick this topic. We had regular meetings. I don't have to look elsewhere for a role model.*

*This work was done jointly with Pranav Garg. Thank you, Pranav, for all the bright ideas, the clarifications for my doubts and for listening out to all my suggestions.*

*The intellectual ambience of the Department of Computer Science and Engineering has ensured that I constantly pursue better understanding of my field. I would like to thank the faculty and the students of the department for maintaining and nurturing this environment.*

*Thanks to my mother, who has been a constant source of inspiration and support.*

Arpita Korwar





# Contents

|          |                                                                   |           |
|----------|-------------------------------------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                                               | <b>1</b>  |
| <b>2</b> | <b>Preliminaries</b>                                              | <b>5</b>  |
| <b>3</b> | <b>Isolation based approach</b>                                   | <b>9</b>  |
| 3.1      | Nice cycles . . . . .                                             | 9         |
| 3.2      | Isolating Weight Assignment in a Planar Bipartite Graph . . . . . | 14        |
| 3.2.1    | Assigning signs to sides of edges. . . . .                        | 15        |
| 3.2.2    | Assigning weights to the edges . . . . .                          | 16        |
| 3.3      | General Planar Graphs . . . . .                                   | 19        |
| 3.3.1    | Cycle basis in planar graph . . . . .                             | 20        |
| 3.3.2    | Finding Basis Edges . . . . .                                     | 22        |
| 3.3.3    | Using basis edges . . . . .                                       | 22        |
| <b>4</b> | <b>Conclusion</b>                                                 | <b>25</b> |

# Chapter 1

## Introduction

*Matching* is one of the most interesting and well-studied combinatorial problems. Edmonds [Edm65] presented the first polynomial time algorithm for maximum matching. In this pioneering work, he was one of the first to suggest that a polynomial time algorithm is efficient. The problem of matching is thus closely related to the theory of complexity. Researchers from various fields e.g. algebra, randomization, combinatorics, complexity theory and algorithms have been trying to find simpler, faster and better algorithms capable of solving the problem of matching.

Given a graph  $G = (V, E)$ , a *matching*  $M \subseteq E$  is a subset of the edges such that no two edges in the matching  $M$  share a common vertex. A *perfect matching* is a matching such that exactly one edge of the matching is incident on each vertex of the graph.

Matching theory has many applications. The theory of perfect matching has been applied in chemistry. The combinatorial problem of *Chinese postman* can be solved using minimum weight perfect matching. More applications of matching theory can be found in the text by Lovasz and Plummer [LP86, Preface] and in the text by Karpinski and Rytter [KR98, Chapters 14 and 15].

The present work deals with the parallel computational model. The complexity class **NC** represents the class of problems that can be solved efficiently by a parallel computer, say parallel, random-access machine (PRAM). The problems in the complexity class **NC** can be solved by a parallel computer with polynomial number ( $n^{O(1)}$ ) of processors in **polylog** ( $\log^{O(1)} n$ ) steps/time.

$\text{NC}^i$  is the class of problems solvable in time  $O(\log^i n)$  on a parallel computer with a polynomial number of processors. Hence,  $\text{NC} = \cup_{i=0}^{\infty} \text{NC}^i$ .  $\text{NC}$  is a subset of  $\text{P}$  because parallel computers can be simulated by sequential ones. It is unknown whether  $\text{NC} = \text{P}$ , though it is widely believed to be false. This means that there are probably some tractable problems which are “inherently sequential” and cannot significantly be sped up by using parallelism.

Given a graph, the following questions may be asked with respect to matching. Does a perfect matching exist in that graph (*existence*)? How many perfect matchings does it contain (*counting*)? Find one perfect matching (*search*) of that graph. We already know that there exists a polynomial time algorithm that *searches* for a perfect matching [Edm65] in a general graph. However, the *counting* problem for bipartite graphs is  $\#P$ -complete [Val79]. The  $\#P$ -complete problems are harder than the class of problems in  $\text{NP}$ , which in turn are believed to be harder than the problems in the complexity class  $\text{P}$ . Hence the problem of *counting* is believed to be harder than the problem of *search*. In fact, Toda’s Theorem [Tod91] implies that a  $\text{P}$  Turing machine with an oracle for *counting* the number of matchings contains the entire polynomial hierarchy ( $\text{PH}$ ).

Randomized  $\text{NC}$  ( $\text{RNC}$ ) algorithms that solve the *decision* and the *search* problems in general graphs are known. The determinant of the Tutte matrix of a graph is a polynomial in  $O(n^2)$  variables. This polynomial is identically zero if and only if there are no perfect matchings in that graph. Also, the Schwartz-Zippel lemma bounds the probability that a non-zero polynomial will have roots at random test points. Random values are assigned to the variables in the Tutte matrix and its determinant is evaluated. Since determinant evaluation is in  $\text{NC}$  [MV99], the *decision* problem can be solved in  $\text{RNC}$ . Using the isolation lemma, weights could be assigned to the edges of the graph such that, with constant probability, a unique perfect matching has the minimum weight. Tutte matrix could be used together with the isolation lemma to obtain an  $\text{RNC}$  algorithm that finds a perfect matching in general graphs [MVV87].

In the case of parallel algorithms for planar graphs, the *searching* problem seems harder than the problem of *counting*. There exists a  $\text{NC}$  algorithm for counting the number of matchings [Kas67]. But no  $\text{NC}$  algorithm is known that would solve the *search* problem for planar graphs. Since *counting* is more difficult than *search* in the sequential case, and since the *counting* problem has

already been solved in the parallel case, it is widely believed that an NC algorithm for *searching* a perfect matching in planar graphs exists.

The first NC algorithm that *searched* for a perfect matching in a bipartite planar graph was presented by Miller and Naor [MN95]. This algorithm was based on network flows in graphs. Mahajan and Varadarajan suggested a linear programming based NC algorithm [MV00] that searched for a perfect matching in bipartite graphs with small genus. Recently, Kulkarni, et al. [DKR08] suggested an NC algorithm which resembled the approach used by isolating lemma.

The problem of *search* in NC has been solved for some special graphs: graphs with **polylog** non-tree edges [LL99], regular bipartite graphs [LPV81] and graphs with **poly** number of matchings [AHT07], where one can list all the perfect matchings.

## Overview of the thesis

In the present work, an attempt has been made to prove that the *search* problem for planar graphs is in NC. One approach to solve the *search* problem is the isolation lemma. With slight modification to the algorithm presented by Mulmuley, Vazirani and Vazirani [MVV87], we can find the edges that are present in at least one perfect matching with the minimum weight. We can find a subset of the edges such that its cardinality is a constant fraction of the total number of edges. These edges are present in all nice cycles. One can probably assign non-zero weights only to these edges to obtain a unique perfect matching with minimum weight. As already discussed, Kulkarni, et al. proposed an NC algorithm to isolate a unique perfect matching with the minimum weight when the input is a planar bipartite graph. They made the input graph Eulerian by adding  $O(n)$  vertices and edges. They then assigned isolating weights to the edges. In our work, this algorithm is made simpler. We do not need to make the graph Eulerian.

The algorithm presented by Kulkarni, et al. relies heavily on the fact that all the cycles in a bipartite graph are even. We look at a possible generalization in the planar graphs case by considering the even cycle subspace.



## Chapter 2

# Preliminaries

In this chapter we develop the background required for the rest of the thesis.

**Definition 2.1** (Graph). *A graph  $G(V, E)$  is an ordered set of vertices  $V$  and edges  $E$ . Each edge is an unordered pair of vertices and is said to be incident on those two vertices.*

A graph can be drawn on a paper with the vertices represented as dots and the edges represented by an arc between the two vertices that it is incident on. The edges may cross each other.

**Definition 2.2** (Planar graph). *A graph which can be drawn on the plane or the sphere without the edges crossing is known as a planar graph.*

Such a drawing of a planar graph is known as a planar embedding of the graph. A graph may have many planar embeddings. As we have seen in the introduction, some matching problems have been solved only for bipartite graphs, and not general graphs. Thus, in a sense, bipartite graphs are simpler than general graphs.

**Definition 2.3** (Bipartite graph). *A graph in which the vertices can be partitioned into two sets such that no two vertices within the same set are adjacent is known as a bipartite graph.*

In a bipartite graph, all cycles of the graph are even. One can refer any standard text on graph theory for a proof of this, e.g. *Graph theory* by Reinhard Diestel [Die05, Proposition 1.6.1.].

**Definition 2.4** (Matching). *Given a graph  $G = (V, E)$ , a matching  $M \subseteq E$  is a subset of the edges such that no two edges share a common vertex.*

A maximum of 1 edge of the matching is incident on any vertex of the graph. In a perfect matching, all the vertices are saturated.

**Definition 2.5** (Perfect matching). *A perfect matching is a matching such that exactly one edge of the matching is incident on each vertex of the graph.*

In this work, the term ‘matching’ would mean a perfect matching, unless specified otherwise.

Let  $G = (V, E)$  be a graph. Then  $n$  and  $m$  represent the number of vertices and the number of edges of that graph respectively. The alphabet  $f$  has been used to refer to the number of faces as well as a particular face in a planar embedding. A planar embedding of graph  $G$  is represented by  $\bar{G}$ . We have used the term ‘graph’ to mean a connected, planar graph. A graph which may be non-planar is referred as ‘general graph’. The term ‘cycle’ has been used to mean a simple cycle, as well as an edge-disjoint union of cycles.

## Tutte matrix

Tutte matrix is a basic tool in matching theory [KR98, Chapter 6]. It has been used to decide whether a perfect matching exists, for counting the number of matchings that a planar graph has and for identifying the minimum weight perfect matching when one exists.

**Definition 2.6** (Tutte matrix). *Given a graph  $G$ , its Tutte matrix  $A$  can be represented as:*

$$A_{ij} = \begin{cases} x_{ij} & \text{if edge } (i, j) \text{ between vertices } i \text{ and } j \text{ exists and } i < j, \\ -x_{ij} & \text{if edge } (i, j) \text{ between vertices } i \text{ and } j \text{ exists and } i > j, \\ 0 & \text{otherwise.} \end{cases}$$

The Tutte matrix is thus a skew-symmetric matrix. The determinant of the Tutte matrix is a polynomial in the variables  $x_{ij}$  and each variable  $x_{ij}$  represents an edge of the graph.

The determinant of any square matrix  $A_{(n \times n)}$  is given by:

$$\det(A) = \sum_{\sigma \in S_n} \text{sgn}(\sigma) \prod_{i=1}^n A_{i, \sigma(i)}$$

where  $S_n$  is the symmetric group on the set  $\{1, 2, \dots, n\}$ . Each term of the determinant corresponds to a unique union of vertex-disjoint even cycles (the length of these cycles may be two) that covers all the vertices of the graph.

## Isolation of a perfect matching

The problem of searching for a perfect matching becomes complex because the graph may contain exponential number of perfect matchings [MR95, Section 12.4.]. A well-known technique that overcomes this complication is based on *the isolation lemma* [MVV87]. Let  $w : S \rightarrow \{1, \dots, r\}$  be a positive integer weight function from the elements in the set  $S$  to  $\{1, \dots, r\}$ . Then the weight of a subset  $R$  of the set  $S$  is the sum of the weights of all the elements in the subset  $R$ .

**Lemma 2.7** (Isolation Lemma). *Let  $S$  be any finite set and  $\mathcal{F}$  be a family of subsets of the set  $S$ . Consider a random weight assignment which assigns every element of set  $S$  a weight uniformly at random from  $\{1, \dots, r\}$ . The probability that there is a unique minimum weight set in  $\mathcal{F}$  is greater than  $\left(1 - \frac{|S|}{r}\right)$ . i.e.*

$$\Pr[\text{unique set with the minimum weight exists}] \geq 1 - \frac{|S|}{r}.$$

To apply this lemma to the perfect matching problem, let the set  $S$  be the set of edges of the graph  $G$ . Hence,  $|S| = m$ . Let the subsets  $S_i$ 's be the matchings of the graph  $G$ . We assign random weights to the edges from the set  $\{1, \dots, 2m\}$ .

The *weight of a matching* is the sum of the weights of all its edges.

$$w_M = \sum_{e \in M} w_e,$$



where  $w_e$  is the weight of the edge  $e$  and  $w_M$  is the weight of matching  $M$ . We try to isolate the matching with the minimum weight. However, a unique minimum weight matching does not always exist.

**Definition 2.8** (Isolating weight assignment). *A weight assignment to the edges is isolating if there exists a unique perfect matching  $M^*$  such that  $w_{M^*} < w_M$  for every matching  $M$  different from  $M^*$ .*

An isolating weight assignment always exists. Suppose the edges in a perfect matching have weights 1 and all other edges have weights 2. Then, any other matching would have weight greater than  $n/2$ . The difficulty lies in finding one such weight assignment deterministically.

According to the isolation lemma, the probability that a random weight assignment is isolating is greater than  $\frac{1}{2}$ . The RNC algorithm by Mulmuley, Vazirani and Vazirani works by assigning weights to the edges in the manner described above [MVV87].

Assuming that the weight assignment is isolating, the minimum weight matching can be found by assigning the Tutte matrix entry for edge  $e$  with weight  $w_e$  to be  $2^{w_e}$  [MVV87]. Let  $M$  be the minimum weight matching. Then the smallest power of 2 that divides the determinant of the Tutte matrix,  $\det(A)$  is  $2^{2w_M}$ . It can be determined whether each edge belongs to this minimum weight matching in  $\text{polylog}$  time.

One possibility of obtaining an NC algorithm for matching is to derandomize the Isolation lemma, i.e. find a weight assignment in NC which leads to a unique minimum weight matching. In the next chapter we describe how this can be done for planar bipartite graphs. Our approach is a simplification of the algorithm presented by Kulkarni, et al. [DKR08]. We also discuss how the approach may be generalized to planar graphs.

## Chapter 3

# Isolation based approach

Recently, Kulkarni, et al. presented an NC algorithm that deterministically found a perfect matching in a planar bipartite graph [DKR08]. The approach is the same as that of the randomized NC algorithm provided by Mulmuley, Vazirani and Vazirani [MVV87], but now an isolating weight assignment is provided deterministically. We try to generalize this idea to the case of general planar graphs. The algorithm by Kulkarni, et al. is based on the fact that all cycles of a bipartite graph are even. Hence, when working with general graphs, we consider the even cycles of the general graph. We also present a simpler version of the algorithm provided by Kulkarni, et al. that works for planar bipartite graphs.

### 3.1 Nice cycles

We have to obtain an isolating weight assignment for the given graph. Nice cycle may be a tool that can be used for that purpose [LP86, Section 8.3.].

**Definition 3.1** (Nice cycle). *A simple cycle  $C$  is called a nice cycle if it is an even cycle and the graph obtained by deleting the vertices of  $C$ ,  $G \setminus V(C)$  has at least one perfect matching.*

The union of two matchings,  $M_1$  and  $M_2$ ,  $M_1 \cup M_2$  leads to a disjoint union of nice cycles and stand-alone edges. The stand-alone edges are part of both matchings -  $M_1$  and  $M_2$ . There exists

at least one nice cycle in  $M_1 \cup M_2$  when  $M_1 \neq M_2$ . Each nice cycle in  $M_1 \cup M_2$  has alternate edges in  $M_1$  and in  $M_2$ .

**Lemma 3.2.** *Given a weight assignment for a graph, if for all nice cycles of the graph the two sets of alternate edges don't have equal weights then that weight assignment is isolating [DKR08]. Equivalently, if  $\sum_{i=1}^r (-1)^i w_{e_i} \neq 0$  where  $e_1, \dots, e_r$  are the successive edges of a nice cycle, then the weight assignment is isolating.*

*Proof.* Let  $S_{C1}$  be one set of alternate edges in the nice cycle  $C$  and let  $S_{C2}$  be its another set of alternate edges.

We have to prove that when a weight assignment provides unequal weights to the two alternating edge sets  $S_{C1}$  and  $S_{C2}$  for all nice cycles  $C$ , the weight assignment is isolating.

Suppose the weight assignment is not isolating. Then there exist at least two perfect matchings with the minimum weight,  $M_1$  and  $M_2$ . Suppose  $M_1 \cup M_2$  leads to  $k$  disjoint nice cycles  $C_1, C_2, \dots, C_k$ , each of length  $> 2$ . Since the weights of the two alternating edge sets in nice cycle  $C_i, 1 \leq i \leq k$  are not equal, i.e. each of the nice cycles  $C_i$  differentiates between its two sets of alternating edges, we can build a matching  $M$  by choosing the set of alternating edges with minimum total weight from each nice cycle  $C_i$ :

$$M = \left\{ \bigcup_{1 \leq i \leq k} \operatorname{argmin} \{wt(S_{C_{i1}}), wt(S_{C_{i2}})\} \right\} \cup \{M_1 \cap M_2\}$$

Since, the matching  $M$  takes the minimum alternating edges from each cycle in the union of matchings  $M_1$  and  $M_2$ ,  $M_1 \cup M_2$ ,  $M_1$  and  $M_2$  could not have both been minimum weight matchings.  $\square$

Let us call the absolute value of the difference between the weights on alternate edges of nice cycle  $C$ ,  $|w_{C1} - w_{C2}|$  as the *alternating weight of cycle  $C$* . From Lemma 3.2 we know that to obtain an isolating weight assignment, we only have to ensure that in each nice cycle of the graph, the alternating weight should be non-zero.

We will try to express the alternating weight constraints of all the nice cycles in the graph as linear inequalities.

Before that, let us look at a few basics.

The *edge space* of a graph  $G$  over a field  $K$  is the  $|E|$ -dimensional vector space over  $K$  where the base elements are indexed by the edges of graph  $G$ . Let  $e_1, e_2, \dots, e_m$  be the edges of the graph  $G$ . Then the edge space consists of formal  $K$ -linear combination

$$\sum x_i \underline{X_{e_i}},$$

where,  $x_i \in K$  and  $\underline{X_{e_i}}$  is the  $|E|$ -dimensional vector with a 1 at the index of edge  $e_i$  and a 0 in all other locations. Henceforth, we will use the field  $\mathbb{F}_2$ , i.e.  $K = \mathbb{F}_2$ . Thus, each vector in the edge space represents a subset of the edges.

The *cycle space* is a subspace of the edge space and is generated by all the simple cycles of the graph. In a general graph, the basis of the cycle space can be obtained by finding a spanning tree; each non-tree edge represents a basis element. This point onwards, we restrict our attention to planar graphs. In a planar graph, all the faces of a planar embedding of the graph are the basis of this subspace. In the cycle space, each of these basis cycles (faces) can be represented as:

$$f = \sum_{i=1}^m b_i \cdot \underline{X_{e_i}},$$

where,

$$b_i = \begin{cases} 1 & \text{if edge } e_i \text{ lies on face } f, \\ 0 & \text{otherwise.} \end{cases}$$

A cycle  $C$  of the planar graph can be expressed as:

$$C = \sum_{j=1}^f d_j \cdot f_j, \tag{3.1}$$

where,

$$d_j = \begin{cases} 1 & \text{if face } f_j \text{ lies in the interior of cycle } C, \\ 0 & \text{otherwise.} \end{cases}$$

These will generate *all* the cycles of the graph. But observe that a cycle can be written as the sum of the faces in its interior, as well as the sum of the faces in its exterior. Hence we exclude the infinite face from the set of basis cycles. There are  $f$  faces. Hence, this leads to  $(f - 1)$  cycle space basis. i.e. the cycle space has  $(f - 1)$  base vectors.

The *even cycle space* is a subspace of the cycle space of a graph. It consists of all even-length cycles. Let

$$n_j = \begin{cases} 1 & \text{if face } f_j \text{ is an odd cycle,} \\ 0 & \text{if face } f_j \text{ is an even cycle.} \end{cases}$$

Then, the even cycle space is given by all vectors  $C_{even} = \sum_{j=1}^f d_j \cdot f_j$  such that

$$\sum_{j=1}^f d_j \cdot n_j = 0 \pmod{2}. \quad (3.2)$$

The solution to this linear equation, the set of all even cycles, forms a kernel of the linear mapping on the left hand side of Equation 3.2. Because of the constraint in Equation 3.2, the dimension of the even-cycles space is one less than the dimension of the general cycle space of the planar graph. There are  $(f - 1) - 1 = (f - 2)$  even cycle basis elements. In a bipartite graph, all cycles are even, which means that all  $n_j$ s are 0. Hence, Equation 3.2 does not reduce the dimension of the cycle space. The dimension of the even cycle space of a bipartite planar graph is still  $(f - 1)$ .

We will now describe a method that may lead to an isolating weight assignment. Indeed, in the case of bipartite planar graphs, it does lead to an isolating weight assignment.

Let us call the basis of the even cycle space as *even cycle basis*. Fix an even cycle basis,  $\{b_1, b_2, \dots, b_k\}$  for the graph  $G$ . For each basis  $b_i$ , fix a linear form  $l_i(\underline{X}) \in \mathbb{Z}[\underline{X}]$  in variable vector  $\{X_e\}_{e \in E}$ . Having fixed the basis and the linear forms  $l_i(\underline{X})$ , associate for any even cycle

$C = \sum_{i=1}^k x_i \cdot \underline{b}_i$ , where  $x_i \in \{0, 1\}$ , the linear form  $l_C(\underline{X}) \in \mathbb{Z}[\underline{X}]$ :

$$l_C(\underline{X}) = \sum_{i=1}^k x_i \cdot l_i(\underline{X}).$$

**Theorem 3.3.** *Let the linear forms  $l_1(\underline{X}), l_2(\underline{X}), \dots, l_k(\underline{X})$  be such that*

1. *If edge  $e$  is present in both the basis  $b_i$  and  $b_j$ , then the sum of their linear forms,  $l_i(\underline{X}) + l_j(\underline{X})$  has coefficient 0 for the variable  $X_e$ .*
2. *For any nice cycle  $C$ , the linear form  $l_C(\underline{X})$  should be of the form:*

$$l_C(\underline{X}) = \sum_{i=1}^r (-1)^i \cdot X_{e_i}, \quad (3.3)$$

*where  $e_1, e_2, \dots, e_r$  are the successive edges of the nice cycle  $C$ .*

*Then, any weight assignment  $X_e = w_e$ , such that  $l_i(\underline{w}) > 0$  for all  $i$  is an isolating weight assignment.*

*Proof.* Let  $e_1, e_2, \dots, e_r$  be the successive edges of the nice cycle  $C$ . Recall from Lemma 3.2 that if  $\sum_{i=1}^r (-1)^i \cdot w_{e_i} \neq 0$  then the weight assignment is isolating. The linear form for a nice cycle (Equation 3.3) is given by  $\sum_{i=1}^r (-1)^i \cdot X_{e_i}$ . Hence, the linear form  $l_C(\underline{X})$  exactly represents the requirement for a weight assignment to be isolating.

But, the linear form of the cycle is given by  $l_C(\underline{X}) = \sum_{i=1}^k x_i \cdot l_i(\underline{X})$ . When we solve the inequalities  $l_i(\underline{w}) > 0$ , it ensures that the alternating weight of nice cycle  $C$  is non-zero.  $\square$

The above theorem has not used any property specific to planar graphs and hence works for general graphs.

Given a general graph  $G$ , one should find the

- basis cycles and
- sign and weight for each edge with respect to each basis cycle that it is in,

such that the rules given above are followed.

Before proceeding further, let us look at how we can use cycle basis to find a perfect matching in planar bipartite graphs.

### 3.2 Isolating Weight Assignment in a Planar Bipartite Graph

In this section, we have a simplified version of an algorithm given by Kulkarni, et al. [DKR08]. In that paper, they had converted the planar graph into an Eulerian graph and then assigned weights to the graph edges. However, as we shall see, there is no need to make the graph Eulerian. We demonstrate the use of Theorem 3.3 to find an isolating weight assignment in bipartite planar graphs. The algorithm works in NC.

Consider a planar embedding  $\tilde{G}$  of a planar bipartite graph  $G$ . Let  $m, n, f$  be the number of its edges, vertices and faces respectively. Since graph  $G$  is planar,  $f \leq 2(n - 2)$ . i.e.  $f = O(n)$ . Select one of the faces as the *outer face*. Any simple cycle of the graph  $G$  can be canonically written as the sum of its internal faces. Hence *the basis cycles of a planar bipartite graph are the faces of that graph*. We now describe a method to assign signs and weights to the edges with respect to the basis cycles that they lie on. This sign and weight assignment should follow the rules given in Theorem 3.3.

Any edge  $e$  in graph  $G$  separates two faces of the planar embedding  $\tilde{G}$ , say faces  $F_1$  and  $F_2$ . Consider the edge  $e$  to have two sides -  $e_{F_1}$  and  $e_{F_2}$  in faces  $F_1$  and  $F_2$  respectively. We assign signs to each side of the edge. Let  $\text{sgn}(e_{F_1})$  and  $\text{sgn}(e_{F_2})$  be the signs assigned to these edge sides respectively.

The above definition can be generalized to the sign of an edge with respect to a cycle  $C$ ,  $\text{sgn}(e_C)$ . Each edge is on a cycle because one of the faces that contain it is in the interior of that cycle. The sign of the edge with respect to cycle  $C$  is the same as its sign with respect to the interior face.

Associate with each face  $F$  the following linear form:

$$l_F(\underline{X}) = \sum_{e \in F} \text{sgn}(e_F) \cdot X_e.$$

It now remains to show the assignment of signs and solving of the inequalities  $l_F(\underline{X}) > 0$  such that the rules of Theorem 3.3 are satisfied.

We assign signs to the edges such that two conditions are satisfied:

1. The two sides of each edge get opposite signs. This ensures that the sum of two linear forms,  $l_i(\underline{X}) + l_j(\underline{X})$  does not contain any edges present in both the basis cycles  $F_i$  and  $F_j$ .
2. The alternating edges in each face get opposite signs. We will show that this ensures that the linear form of nice cycle  $C$  represents the alternating weight of nice cycle  $C$ .

### 3.2.1 Assigning signs to sides of edges.

The vertices of a planar embedding of planar graph  $G$ ,  $\bar{G}$  are divided into two colors, say  $A$  and  $B$ . For all vertices  $v \in A$ , we assign signs to the edges incident on  $v$  in the following manner: Imagine standing on the vertex  $v$ . For each edge incident on  $v$ , face the direction in which it is going out of  $v$ . Assign  $+$  sign to the right side of the edge and  $-$  sign to the left side. *Hence, the two sides of each edge are assigned opposite signs.*

Because of this assignment, when we stand at any vertex in  $B$ , the right side of each edge incident on that vertex gets a  $-$  sign and the left side of each edge incident on that vertex gets a  $+$  sign.

Since the vertices in  $A$  can never be adjacent, there is no conflict of sign assigned to an edge side. Since each edge has one end-point in  $A$ , every edge-side gets a sign assigned to it.

**Lemma 3.4.** *The alternate edges of a face  $F$  get opposite signs with respect to that cycle.*

*Proof.* We start travelling in clockwise direction from a vertex along the edges in face  $F$ . Without loss of generality, suppose that the vertex has color  $A$ . At each vertex, the signs of the edges  $e_i$  with respect to face  $F$  is equal to the signs assigned to the right hand side of the edges when standing at a vertex of that particular color. When we travel along the face, we alternately visit vertices with colors  $A$  and  $B$ . Hence the alternate edges get opposite signs with respect to that face.  $\square$

**Lemma 3.5.** *The alternate edges of a simple cycle  $C$  get opposite signs with respect to that cycle.*



*Proof.* Proof is the same as above. □

Thus, the alternating edges in all nice cycles are assigned opposite signs.

We have now found an even cycle basis and the corresponding linear forms such that the rules of Theorem 3.3 are followed.

### 3.2.2 Assigning weights to the edges

We now present an algorithm that solves the inequalities  $l_i(\underline{X}) > 0$  in NC. We will ensure that the alternating weight of each face is 1.

**Definition 3.6** (Dual tree). *Let  $\bar{G}$  be a planar embedding of graph  $G$ . Let  $\bar{H}$  be its dual graph. Then a spanning tree  $T_{dual}$  of the dual graph  $\bar{H}$  is called as a dual tree.*

Suppose edge  $e$  separates faces  $F_1$  and  $F_2$  in the planar embedding  $\bar{G}$  of graph  $G$ . Then the edge  $e$  connects the vertices representing the faces  $F_1$  and  $F_2$  in the dual graph. Thus, the dual tree may be seen as a spanning tree on the faces of graph  $\bar{G}$ . Root the dual tree at any face.

We will ensure that the alternating weight of each face in graph  $\bar{G}$  is 1. The algorithm that provides these weights runs in NC.

We assign weight 0 to all the edges that don't belong to the dual tree  $T_{dual}$ . The edges in  $T_{dual}$  are assigned non-zero weights. At a leaf node that represents face  $F$  of the graph  $\bar{G}$ , only the edge that connects face  $F$  to its parent is allowed a non-zero weight. Hence, only the weight of this edge should be manipulated so that the alternating weight of face  $F$  is 1. Let  $e(F)$  be the unique edge in the dual tree  $T_{dual}$  which connects node  $F$  to its parent. Then, once the weights of the edges that connect the children of face  $F$ ,  $w(e(f))$  are fixed, the weight of the edge that connects face  $F$  to its parent,  $w(e(F))$  is the only value that is yet to be fixed. This edge is given a weight such that the alternating weight of face  $F$  is 1.

But this method may take  $O(n)$  steps (e.g. when the tree is degenerate). We want an algorithm that runs in  $\log^{O(1)}(n)$  time. Henceforth in this section, let  $n$  be the number of faces in graph  $\bar{G}$ . This number equals the number of nodes in the dual tree.

We now look at an algorithm that runs in  $\text{NC}^2$  steps and assigns weights to the edges simultaneously. Let  $n(F)$  be the number of vertices in the subtree rooted at node  $F$ . Let  $\text{children}(F)$  be the set of the children of node  $F$  in the dual tree  $T_{\text{dual}}$ . Since face  $F$  and the faces  $f \in \text{children}(F)$  are neighbors in graph  $\tilde{G}$  (face  $F$  and face  $f$  share edge  $e(f)$ ),  $\text{sgn } e_F(f) = -\text{sgn } e_f(f)$ .

**Lemma 3.7.** *If, for all faces, the weight assignment is*

$$w(e(F)) = \text{sgn}(e_F(F)) \cdot n(F),$$

*then each face gets alternating weight = 1.*

*Proof. Base case:* When  $F$  is a leaf node, the number of nodes in the subtree rooted at  $F$ ,  $n(F) = 1$ .

We want:

$$\sum_{e \in F} \text{sgn } e_F \cdot w(e) = 1$$

But the non-tree nodes are assigned weight 0. Only the edge  $e(F)$  can be assigned a non-zero weight. Hence,

$$\text{sgn } e_F(F) \cdot w(e(F)) = 1$$

Hence,

$$w(e(F)) = \text{sgn } e_F(F) \cdot 1$$

Now, to prove the induction step.

**Induction assumption:** For each child node  $f_i \in \text{children}(F)$ , we assume that the weight of the edge connecting face  $f_i$  to its parent face  $F$  is given by:  $w(e(f_i)) = \text{sgn}(e_{f_i}(f_i)) \cdot n(f_i)$ .

**Inductive step:** We want the alternating weight of face  $F$  to be 1.

$$\begin{aligned}
\sum_{e \in F} \text{sgn } e_F \cdot w(e) &= 1, \\
\text{sgn } e_F(F) \cdot w(e(F)) + \sum_{f_i \in \text{children}(F)} \text{sgn } e_F(f_i) \cdot w(e(f_i)) &= 1, \\
\text{sgn } e_F(F) \cdot w(e(F)) + \sum_{f_i \in \text{children}(F)} (-1) \text{sgn } e_{f_i}(f_i) \cdot (\text{sgn } e_{f_i}(f_i) \cdot n(f_i)) &= 1, \\
\text{sgn } e_F(F) \cdot w(e(F)) - \sum_{f_i \in \text{children}(F)} n(f_i) &= 1,
\end{aligned}$$

Hence,

$$\begin{aligned}
w(e(F)) &= \text{sgn } e_F(F) \cdot \left( 1 + \sum_{f_i \in \text{children}(F)} n(f_i) \right), \\
w(e(F)) &= \text{sgn } e_F(F) \cdot n(F).
\end{aligned}$$

Since faces  $F$  and  $f_i$  are neighbours sharing edge  $e(f_i)$ ,  $\text{sgn } e_F(f_i)$  is replaced with  $(-1) \text{sgn } e_{f_i}(f_i)$ . From the induction step,  $w(e(f_i))$  is replaced with  $\text{sgn } e_{f_i}(f_i) \cdot n(f_i)$ . The number of vertices in the subtree rooted at face  $F$  is given by  $\left( 1 + \sum_{f_i \in \text{children}(F)} n(f_i) \right)$ . Hence, if we assign weights to the edges according to the above statement, the alternating weight of each face is 1.  $\square$

It now remains to prove that  $n(F)$  can be found in NC.

**Lemma 3.8.** *The number of vertices in a subtree rooted at node  $F$ ,  $n(F)$  can be counted in NC<sup>2</sup>.*

*Proof.* Consider the orientation  $\bar{T}_{dual}$  of the undirected dual tree  $T_{dual}$  in which each edge is oriented from the parent to the child. Then the vertices in a subtree are exactly the vertices reachable from the root of that subtree. The problem of finding the number of vertices in a subtree thus reduces to finding the set of vertices reachable from each node (the REACHABILITY problem), and then counting the cardinality of each set. The REACHABILITY problem is in NC<sup>2</sup> and works with

$O(n^3)$  processors [Pap93, Section 15.1]. The output of the REACHABILITY problem is a vector in  $\{0, 1\}^n$ , one for each of the vertices. The addition of  $n$  numbers takes  $O(\log n)$  steps and uses  $O(n)$  processors [Pap93, Section 15.1]. Hence,  $n(F)$  can be counted in  $\text{NC}^2$ .  $\square$

Hence, the algorithm for an finding an isolating weight assignment is given by algorithm 1.

---

**Algorithm 1** OBTAINING AN ISOLATING WEIGHT ASSIGNMENT

---

**Require:** A planar bipartite graph  $G = (A, B)$ .

- 1: Obtain a planar embedding  $\tilde{G}$  of graph  $G$  [KR88]. Fix an outer face.
  - 2: **for all** edges  $e(F) \in \tilde{G}$  **do in parallel**
  - 3:   Let edge  $e$  be incident on vertex  $v \in A$  and separate faces  $F_1$  and  $F_2$  of  $\tilde{G}$ . Let  $F_2$  appear immediately after  $F_1$  when travelling clockwise along the edges incident on vertex  $v$ . Then,  $\text{sgn } e_{F_1} \leftarrow -1$  and  $\text{sgn } e_{F_2} \leftarrow +1$ .
  - 4: **end for**
  - 5: Find a spanning tree  $T$  of  $G$  [GR89]. Assign weight 0 to all the edges in  $T$ .
  - 6: Construct the dual graph  $\tilde{H}$  of  $\tilde{G}$ . The faces of  $\tilde{G}$  are the vertices of  $\tilde{H}$ .
  - 7: Delete the edges from  $\tilde{H}$  that correspond to edges in  $T$ . The resultant graph  $T_{dual}$  is a tree. Let the vertex corresponding to the outer face be the root of this tree.
  - 8: **for all** nodes  $F \in T_{dual}$  **do**
  - 9:   Find the vertices in the subtree rooted at node  $F$  using the parallel algorithm for REACHABILITY [Pap93, Section 15.1].
  - 10: **end for**
  - 11: **for all** nodes  $F \in T_{dual}$  **do**
  - 12:    $n(F) \leftarrow$  the number of vertices in the subtree rooted at node  $F$  obtained by using the parallel algorithm for addition of  $n$  numbers [Pap93, Section 15.1].
  - 13: **end for**
  - 14: **for all** edges  $e(F) \in T_{dual}$  **do in parallel**
  - 15:    $w(e(F)) \leftarrow \text{sgn}(e_F(F)) \cdot n(F)$ .
  - 16: **end for**
- 

The algorithm for constructing the dual tree has been taken from a text by Karpinski and Rytter [KR98, Section 13.3].

Each step of algorithm 1 takes  $O(\log^2(n))$  time, except the last step of addition, which can be performed in  $O(\log n)$  time. Hence, the algorithm is in  $\text{NC}^2$ .

### 3.3 General Planar Graphs

We can select  $(f - 2)$  edges of planar graph  $G$  ( $f$  is the number of faces of  $G$ ). These edges are called, say, *basis edges*. The set of basis edges spans all the matchings and all the nice cycles.

### 3.3.1 Cycle basis in planar graph

In a planar graph  $G$ , one would naturally suppose that the faces of a planar embedding of  $G$  would lead us to the basis cycles. We will be proceeding in this direction. Since we want to generate the nice cycles, and since all nice cycles are even, we generalize the situation and generate all the even cycles.

#### Algorithm for generating the even cycle subspace

Consider a planar embedding  $\bar{G}$  of a 2-connected planar graph  $G$ .

We will need the following lemma before we proceed:

**Lemma 3.9.** *The parity of number of odd faces in the interior of a simple cycle  $C$  is the same as the parity of the length of  $C$ .*

*Proof.* Let  $F_C$  denote the set of faces in the interior of  $C$ . Let  $e(f)$  denote the edges of face  $f$ . Let  $E_C$  denote the set of edges in the interior of  $C$ . Each of these interior edges occurs on the boundary of exactly two faces in  $F_C$ . The boundary edges of  $C$  occur in exactly one of the faces in  $F_C$ . Hence,

$$\begin{aligned} \sum_{f \in F_C} e(f) &= 2|E_C| + |C| \\ \sum_{f \in F_C, e(f) \text{ is even}} e(f) + \sum_{f \in F_C, e(f) \text{ is odd}} e(f) &\equiv |C| \pmod{2} \\ \sum_{f \in F_C, e(f) \text{ is odd}} e(f) &\equiv |C| \pmod{2} \\ |\text{odd faces in } F_C| &\equiv |C| \pmod{2}. \end{aligned}$$

□

In particular, a simple even cycle contains even number of faces of odd length (odd faces) in its interior.

Since  $G$  is planar, we can construct its dual graph. The vertices of the dual graph represent the

faces of the primal graph. Hence the vertices in the dual graph can be classified as *even face* or *odd face* according to the number of bordering edges in the primal graph.

The dual tree of a planar graph  $G$ , is a spanning tree of the dual graph of  $G$ . Let us denote it by  $H$ . The algorithm for constructing the dual tree has been taken from a text by Karpinski and Rytter [KR98, Section 13.3].

The planar embedding  $\bar{G}$  of the planar graph  $G$  is embedded on a sphere. We may now consider any one of the faces as the outer face of the planar embedding  $\bar{G}$ .

Algorithm 2 is used for finding the basis  $\mathcal{B}$  of the even cycle space.

---

**Algorithm 2** OBTAINING AN EVEN CYCLE BASIS OF A PLANAR GRAPH

---

**Require:** A planar graph  $G$ .

```

1:  $\mathcal{B} = \{\}$ 
2: Obtain a planar embedding  $\bar{G}$  of  $G$  [KR88].
3: Find a spanning tree  $T$  of  $G$  [GR89]. Assign weight 0 to all the edges in  $T$ .
4: Construct the dual graph  $\bar{H}$  of  $\bar{G}$ . The faces of  $\bar{G}$  are the vertices of  $\bar{H}$ .
5: Delete the edges from  $\bar{H}$  that correspond to edges in  $T$ . The resultant graph  $T_{dual}$  is a tree.
   Let the vertex corresponding to the outer face be the root of this tree. Pick a vertex  $f_{root}$  in  $H$ 
   which corresponds to an odd face. Make it the infinite face of  $\bar{G}$  and root  $H$  at  $f_{root}$ .
6: for all  $f \in V(H), f \neq f_{root}$  do
7:   if  $f$  is an even face then
8:      $\mathcal{B} = \mathcal{B} \cup f$ .
9:   else
10:     $C = f$ .
11:     $f_{current} = f$ .
12:    repeat
13:       $f_{current} = \text{parent of face } f_{current} \text{ in dual tree } H$ .
14:       $C = C + f_{current}$ 
15:    until  $f_{current}$  is odd
16:   end if
17: end for
```

---

The basis  $\mathcal{B}$  contains all the even faces. Also, each odd face is paired up with its closest odd ancestor.

This basis does indeed generate all the simple even cycles. Consider an even cycle  $C$ . It contains even number of odd faces in its interior. We pair up these odd faces arbitrarily. Suppose amongst a pair of odd faces, one is the ancestor of another. Then, when we add all the intermediate cycle basis the intermediate odd faces get added twice. They are eliminated. We get the pair of odd

faces that we wanted. If the two odd faces are in different subtrees, then there exists a common odd ancestor of these two odd faces. We proceed as in the previous case, then add the common ancestor twice, getting rid of it. This algorithm reduces to REACHABILITY and hence, is in  $\text{NC}^2$ .

We'd next find some representative edges of the basis cycles. These edges would be present in exactly one even cycle basis.

### 3.3.2 Finding Basis Edges

We'd like to find a set of edges such that each nice cycle contains at least one of these edges. Hence, now we can assign non-zero weights only to these edges and still get an isolating weight assignment.

Create an  $(f - 1) \times m$  boolean matrix  $A$  with the matrix element  $a_{ij} = 1$  if edge  $e_j$  lies on the boundary of face  $i$ . Perform Gaussian elimination on matrix  $A$ . This can be performed in  $\text{NC}$  [BDHM92] and leads to  $(f - 2)$  even cycle basis elements. Since we have performed Gaussian elimination, we get a new set of even cycle basis. Each of them has a unique 'representative edge'. We call these edges the *basis edges*.

### 3.3.3 Using basis edges

**Lemma 3.10.** *Any nice cycle is the sum of the even-cycle basis elements represented by the basis edges on its boundary.*

*Proof.* Any simple even cycle  $C$  can be represented as a sum of even-cycle basis elements with addition of the elements in  $\mathbb{Z}/2\mathbb{Z}$ . When we add the vectors in  $\mathbb{Z}/2\mathbb{Z}$ , the basis edges, since they are present only once in the sum, will have value 1 in the sum vector. This means that they are present on the boundary of the resultant cycle  $C$ .

Now, to prove that: If an edge is present on the boundary of nice cycle  $C$ , then the even-cycle basis element it represents is present in the representation of  $C$  as a sum of even-cycle basis. Suppose that even-cycle basis element was not present. Then, the boundary of the nice cycle would not include that edge.  $\square$

Since all the nice cycles are represented by a fraction of the edges, hopefully, we can assign

weights only to these basis edges (other edges are assigned weight 0) and obtain an isolating weight assignment.

This approach generalizes the even cycles of a planar bipartite graph to the even cycles of a general planar graph. We need to find the basis cycles and assign signs to the edges such that the conditions in Theorem 3.3 are satisfied. We have found the basis cycles. But we could not find a NC method to assign signs to the edges. If such a method is found, the problem of finding a matching of planar graphs in NC would be solved.





## Chapter 4

# Conclusion

We have tried to find an NC algorithm that searches for a perfect matching in a planar graph. We tried to derandomize the algorithm based on the Isolation lemma. In the process, we defined even cycle space of a graph and described a method to find its basis in a planar graph (Section 3.3). We found the set of basis edges, which has cardinality equal to a fraction of the total number of edges in the graph.

One could now proceed by assigning isolating weights to the basis edges in logarithmic number of steps. Another approach would be to solve the problem in steps; reduce the number of edges in the input graph to a constant fraction in one step. This would ensure that the algorithm halts in logarithmic number of steps.

We hope that the ideas presented in this work will be useful in finding an efficient parallel algorithm that searches for a perfect matching in a graph.



# Bibliography

- [AHT07] Manindra Agrawal, T M Hoang, and Thomas Thierauf. The polynomially bounded perfect matching problem is in  $\text{NC}^2$ . *LNCS*, 4393:489–499, 2007.
- [AKLS00] Dan Archdeacon, Jin Ho Kwak, Jaeun Lee, and Moo Young Sohn. Bipartite covering graphs. *Discrete Math*, 214:51–63, 2000.
- [BDHM92] G. Buntrock, C. Damm, U. Hertrampf, and C. Meinel. Structure and importance of logspace-MOD classes. *Mathematical Systems Theory*, 25(3):223–237, 1992.
- [Die05] Reinhard Diestel. *Graph Theory (Graduate Texts in Mathematics)*, volume 173. Springer-Verlag, Heidelberg, 3rd edition, July 2005.
- [DKR08] Samir Datta, Raghav Kulkarni, and Sambuddha Roy. Deterministically isolating a perfect matching in bipartite planar graphs. *Symposium on Theoretical Aspects of Computer Science*, pages 229–240, 2008.
- [Edm65] J. Edmonds. Path, trees, and flowers. *Canadian J. Math.*, 17:449–467, 1965.
- [GR89] Alan Gibbons and Wojciech Rytter. *Efficient Parallel Algorithms*. Cambridge University Press, 1989.
- [GT87] Jonathan L. Gross and Thomas W. Tucker. *Topological Graph Theory*. New York: Wiley, 1987.

- [Kas67] P. W. Kasteleyn. *Graph Theory and Theoretical Physics by F. Harary, editor*, chapter Graph theory and crystal physics., page 43110. Academic Press, 1967.
- [KR88] P. Klein and J. Reif. An efficient parallel algorithm for planarity. *Journal of Computer and System Sciences*, 37(2):190–246, 1988.
- [KR98] Marek Karpinski and Wojciech Rytter. *Fast parallel algorithms for graph matching problems*. Oxford lecture series in mathematics and its applications. Oxford University Press, Inc., New York, NY, USA, 1998.
- [Kul06] Raghav Kulkarni. A new NC-algorithm for finding a perfect matching in d-regular bipartite graphs when d is small. *Lecture notes in computer science*, 3998:308–319, 2006.
- [LL99] J. Lakhal and L. Litzler. An NC algorithm for the perfect matching problem in larger cycle-free graphs. *Electronic Notes in Discrete Mathematics*, 3:114–117, May 1999.
- [LP86] L. Lovasz and M.D. Plummer. *Matching Theory (North-Holland mathematics studies)*. Elsevier Science Ltd, 1986.
- [LPV81] G. F. Lev, N. Pippenger, and L. Valiant. A fast parallel routing algorithm for routing in permutation networks. *IEEE Transactions on Computers*, C-30:93100, February 1981.
- [MN95] Gary L. Miller and J. Naor. Flow in planar graphs with multiple sources and sinks. *SIAM J. Comput.*, 24(5):1002–1017, 1995.
- [MR95] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*, chapter 12: Parallel and Distributed Algorithms. Cambridge University Press, The Pitt Building, Trumpington Street, Cambridge, United Kingdom, 1995.
- [MV99] Meena Mahajan and V Vinay. Determinant: old algorithms, new insights. *SIAM Journal on Discrete Mathematics*, 12(4):474–490, 1999.
- [MV00] Meena Mahajan and Kasturi R. Varadarajan. A new NC-algorithm for finding a perfect matching in bipartite planar and small genus graphs (extended abstract). In *STOC '00*:

- Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 351–357, New York, NY, USA, 2000. ACM.
- [MVV87] K. Mulmuley, U.V. Vazirani, and V.V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.
- [Pap93] Christos Papadimitriou. *Computational Complexity*. Addison Wesley, December 1993.
- [Sta83] John R. Stallings. Topology of finite graphs. *Inventiones Mathematicae*, 71:551–565, 1983.
- [Tod91] Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.
- [Val79] Leslie G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science (Elsevier)*, 8:189–201, 1979.