Survey of Algorithms for Different Matchings

A Thesis

submitted to
Indian Institute of Science Education and Research Pune
in partial fulfillment of the requirements for the
BS-MS Dual Degree Programme

by

Mitali Thatte

Indian Institute of Science Education and Research Pune
Dr. Homi Bhabha Road,
Pashan, Pune 411008, INDIA.

May, 2019

Supervisor: Dr Meena Mahajan

Certificate

This is to certify that this dissertation entitled Survey of Algorithms for Different Matchings towards the partial fulfilment of the BS-MS dual degree programme at the Indian Institute of Science Education and Research, Pune represents study/work carried out by Mitali Thatte at Indian Institute of Science Education and Research under the supervision of Dr Meena Mahajan, Professor, Institute of Mathematical Sciences , during the academic year 2018-2019.

[illegible]

Dr Meena Mahajan

Committee:

Dr Meena Mahajan

Dr Soumen Maity

This thesis is dedicated to Saurabh

Declaration

I hereby declare that the matter embodied in the report entitled Survey of Algorithms for Different Matchings are the results of the work carried out by me at the Institute of Mathematical Sciences, Indian Institute of Science Education and Research, Pune, under the supervision of Dr Meena Mahajan and the same has not been submitted elsewhere for any other degree.

[illegible]

Mitali Thatte

Acknowledgments

I would like to thank Dr Meena Mahajan for being a wonderful guide and mentor. I would also like to thank Dr Soumen Maity, The Institute of Mathematical Sciences, IISER Pune and all my friends and family for supporting me.

Abstract

The main aim of this project is to survey some techniques used to develop algorithms to find different types of matchings. The three main papers that we survey are:

1. Bipartite Perfect Matching is in quasi-NC- Stephen A. Fenner, Rohit Gurjar, Thomas Thierauf. This paper gave a technique to derandomize the famous isolation lemma using $(n^{6 \log n})$ processors working in parallel to find a perfect matching in a bipartite Graph

2. The Matching Problem in General Graphs is in Quasi-NC- Ola Svensson, Jakub Tarnawski. This paper built on the tecniques of the previous paper to give a quasi-NC algorithm to find a perfect matching in General graphs.

3. Popular Matchings.- David J. Abraham, Robert W. Irving, Telikepalli Kavitha, Kurt Mehlhorn. This was one of the first papers to give a polynomial time algorithm to find popular matching in bipartite graphs where nodes in one partition have preferences regarding which node in the other partition to be matched to .

Contents

Abstract **xi**

1 Preliminaries **3**

1.1 The Perfect Matching Polytope 3

1.2 RNC Algorithm for Perfect Matching 5

2 Perfect Matching: Bipartite Graphs **13**

3 Pseudo-deterministic Bipartite Perfect Matching **19**

4 Perfect Matching: General Graphs **25**

5 Popular Matchings **31**

5.1 Characterization 31

5.2 The Algorithm 36

5.3 Strict Preferences 37

6 Conclusions **39**

Introduction

Let $G(V, E)$ be a graph with $|V| = n, |E| = m$. A matching M of a graph $G(V, E)$ is a subset of the set of edges satisfying the property that no two edges in M have an endpoint in common. A matching M is said to be maximum if for all matchings M' of G we have $|M| \geq |M'|$. A polynomial time algorithm to find a maximum matching in a graph was given by Edmonds in 1965 [1]. A more efficient algorithm to find a maximum matching for bipartite graphs was given by Hopcraft and Karp in 1978 [2].

A matching which covers every vertex is called a perfect matching. We can see that if a perfect matching exists, Edmond's algorithm will find in in polynomial time. An interesting problem is to find a perfect matching using polynomially many processors working in parallel in poly-log time. This is the class NC. In the papers we surveyed about perfect matchings, the edges of the graphs are given weights so as to enable us to find a perfect matching. The core problem essentially is to find a clever weight function on the set of edges so that the minimum weight perfect matching is unique. The famous Isolation Lemma by Mulmuley, Vazirani and Vazirani [3] gives us a way to find a weight function with high probability. We can isolate the minimum weight perfect matching in the graph if it exists using the Tutte matrix of the graph without using randomization in poly-log time using polynomially many processors. We don't know how to derandomize the isolation lemma to find a perfect matching in NC. Several modified instances have been solved though, for example, if it is known that the number of perfect matchings in a graph is polynomially bounded, then it is possible to compute all of these matchings by a deterministic NC algorithm. One algorithm for the same was given by D Grigoriev and M Karpinski [4] and another by Agrawal, Hoang and Thierauf [5].

Fenner, Gurjar and Thierauf gave a way to derandomize the isolation lemma for bipartite graphs [6] with a slightly extra allowance. They gave a way to find a set of $n^{6 \log n}$ weight functions with weights bounded by $n^{6\log n}$ such that at least one weight function in this set is Isolating (minimum weight perfect matching is unique). We call such an algorithm quasi-NC because the number of processors involved is not polynomial in n but $n^{O(\log n)}$. The important tools they used to find such a weight function were; the perfect matching polytope of bipartite graphs, the fact that all the cycles have even length. They essentially ensured that at least one weight function among these gives non-zero circulation to all the cycles in the graph so that the minimum weight perfect matching is unique. Svensson and Tarnawski [7] dealt with the limitations of the preceding problems to give a quasi-NC algorithm for general graphs. They introduced the notion of a cycle respecting the face of a polytope to deal with cycles which are contained in the symmetric difference of two perfect matchings.

David J. Abraham, Robert W. Irving, Telikepalli Kavitha, and Kurt Mehlhorn [8] gave a characterization for popular matchings in bipartite graphs so as to allow for a polynomial time algorithm to find one. A vertex v is said to prefer a matching M over M' if either it is matched in M but unmatched in M' or if it matched to a vertex w in M and w' in M' with w being strictly higher than w' on v's preference list. v is said to indifferent between M and M' if it is unmatched in both or matched in both to vertices which tie on v's preference list. A matching M is said to be popular if for all matchings M', the number of vertices that prefer M over M' is greater than or equal to the number of vertices that prefer M' over M.

Chapter 1

Preliminaries

1.1 The Perfect Matching Polytope

Let $G(V,E)$ be a graph with $|V| = n, |E| \leq n^2$. A subset of $\mathbb{R}^{|E|}$ is called the perfect matching polytope of the graph G if it is the convex hull of all its perfect matching points. A point $\overrightarrow{\boldsymbol{x}} \in \mathbb{R}^{|E|}$ with entries in $\{0,1\}$ is a representation of a subset $S \subseteq E$ such that

$$x_e = \begin{cases} 1 & \text{if } e \in S \\ 0 & \text{otherwise} \end{cases}$$

$\overrightarrow{\boldsymbol{x}}$ is said to be a perfect matching point representing the perfect matching M if

$$x_e = \begin{cases} 1 & \text{if } e \in M \\ 0 & \text{otherwise} \end{cases}$$

We claim that for a bipartite graph $G(V,E)$, the perfect matching polytope is given by the following constraints.

$$\textstyle\sum_{e \in \delta(v)} x_e = 1 \quad \forall v \in V \tag{1.1}$$

$$x_e \geq 0 \tag{1.2}$$

We can see that any convex combination of some perfect matching points will be vector satisfying the above constraints. It remains to be shown that any point satisfying the above constraints can be expressed as a convex combination of some perfect matching points.
Let $\overrightarrow{\boldsymbol{x}} \in \mathbb{R}^{|E|}$ be a point satisfying the above constraints such that $x_e > 0$ for some subset $S \subset E$. Then the graph $G(V, S)$ is a perfect matching or it contains a cycle.
Suppose not.
Equation 1.1 ensures that every vertex in $G(V, S)$ has degree at least one. Therefore $|S| \geq \frac{n}{2}$. As $G(V, S)$ is not a perfect matching, $|S| > \frac{n}{2}$. Further as $G(V, S)$ does not contain a cycle, there are two vertices v_1, v_2 of degree 1 such that there exist a path of length at least two between v_1 and v_2 say $(v_1 v_3 ... v_2)$. But as v_1 has degree 1, the edge incident on v_1 say e_1 will have corresponding value 1 in $\overrightarrow{\boldsymbol{x}}$. But this means that v_3 cannot have any more neighbors again due to constrain 1.1. Therefore the path of length greater than or equal to two cannot exist.
Therefore $G(V, S)$ is a perfect matching or contains a cycle.

If $\overrightarrow{\boldsymbol{x}}$ is a perfect matching point, then we are done ie, we can obviously express it as a convex combination of itself.
Therefore,we assume that $\overrightarrow{\boldsymbol{x}}$ is not a perfect matching point so that the corresponding set S contains a cycle $C = (e_1 e_2 ... e_{2l})$ where $x_{e_i} > 0$ where $i \in \{1, 2, ..., 2l\}$. Because G is a bipartite graph, C will always be even. Now we consider another vector $\overrightarrow{\boldsymbol{y}}$ such that

$$y_e = \begin{cases} x_e - (-1)^i \epsilon & \text{if } e = e_i \in C \\ x_e & \text{otherwise} \end{cases}$$

Here $\epsilon = \min\{x_{e_1}, x_{e_2}, ... x_{e_{2l}}\}$ We can see that $\overrightarrow{\boldsymbol{y}}$ satisfies constraints 1.1 and 1.2. So now either $\overrightarrow{\boldsymbol{y}}$ is a perfect matching point or corresponds to a set containing a cycle in which case we proceed as before. This process is finite as the edges for which we have made x_e zero will never be given non-zero value again. Therefore we can eventually reach a perfect matching point.This is equivalent to moving towards a vertex of a convex hull of perfect matchings while staying within the hull. Therefore the perfect matching polytope is completely described by 1.1 and 1.2.

1.2 RNC Algorithm for Perfect Matching

An NC algorithm is one where polynomially many processors work in parallel to give an output in poly-log time. It remains an open problem to determine whether there exists an NC algorithm to find a perfect matching. An RNC algorithm has the additional allowance of using polynomially many random bits and should satisfy the following two properties:

1. If the correct answer is 'Yes', it returns 'Yes' with probability $\geq 1/2$. In our context it should return a perfect matching, if one exists, with probability $\geq 1/2$
2. If the correct answer is 'No' it returns 'No' with probability 1. That is, if no perfect matching exists we should be able to get the output; 'This graph has no perfect matching' with probability 1.

It has been shown that both the decision and the search version of the perfect matching problem are in RNC.

1.2.1 Bipartite Graphs - Decision

Let $A_{n\times n}$ be the bipartite adjacency matrix of graph $G(V \cup U, E)$ with $|V| = |U| = n, |E| = m \leq n^2$.

$$a_{ij} = \begin{cases} 1 & \text{if } \exists \text{ an edge between vertices } i \text{ and } j \\ 0 & \text{otherwise} \end{cases}$$

We define a matrix B such that

$$b_{ij} = a_{ij}x_{ij} \tag{1.3}$$

The determinant of this matrix will be

$$|B| = \sum_{\sigma\in S_n} Sign(\sigma) \prod_{i=1}^{n} b_{i\sigma(i)} \tag{1.4}$$

Where $Sign(\sigma) \in \{1, -1\}$ is associated with the permutation σ and equals $(-1)^k$ where k is the number of inversions in the permutation. An inversion is a pair (x, y) such that $x < y$ and $\sigma(x) > \sigma(y)$. We can see that every perfect matching in the graph G can be expressed as a permutation of S_n and every permutation can be associated to a perfect matching. As x_{ij} are indeterminates, no term cancels out any other term in the expansion of determinant of $|B|$. Some of the b_{ij} may be zero but every non-zero term represents one perfect matching in the graph. From this, the next theorem follows easily

Theorem 1.1. *$|B|$ not identically zero $\iff$ G has a perfect matching*

Proof. $\Longrightarrow$ if $|B|$ is not identically zero, then it has a permutation σ as a monomial in its expansion. This term corresponds to the perfect matching $M = \{(i, \sigma(i)) \in E\}$
$\Longleftarrow$ If G has a perfect matching, then the permutation corresponding to this perfect matching, that is $\forall i, \sigma(i) = M(i)$ is present. And as no term gets canceled in the expansion of the determinant, $|B|$ is not identically zero. □

We know that the problem of calculating the determinant of matrix with integer entries is in NC [9] But here the number of monomials in the expansion of $|B|$ may be at most $n!$ which is not polynomially many. x_{ij} being indeterminates, these monomials cannot be further reduced to manageable number of terms. This is where we need to evoke randomization to check of $|B|$ is identically zero. We will need to make use of following result.

Theorem 1.2 (Schwartz Zippel)**.** *Let p be a polynomial of degree n on m variables. Let $Q = \{1, 2, ..., N\}$ such that $N \geq n$. Let $b_i, i \in \{1, 2, .., m\}$ be chosen randomly and independently from the set Q. Let $\overrightarrow{\boldsymbol{b}} = (b_1, b_2, ..., b_m)$. Then*

$$P(p(\overrightarrow{\boldsymbol{b}}) \neq 0) \geq 1 - \frac{n}{N} \tag{1.5}$$

Here $p(\overrightarrow{\boldsymbol{b}})$ is the polynomial p with b_i substituted in place of the m variables.

So now we have an algorithm of check of the bipartite graph G has a perfect matching. We assign values to every x_{ij} independently and randomly from $\{1, 2, ..., 2n\}$ to get the modified matrix B'. It is easy to see that $|B| = 0 \Longrightarrow |B'| = 0$ That is, if our graph does not contain any perfect matching, then $|B'|$ will always be evaluated to zero. However if $|B| \neq 0$ then $|B'|$ will not be evaluated to zero with probability $\geq 1 - \frac{n}{2n} = \frac{1}{2}$. Therefore, if the correct answer is 'Yes', that is if G does contain a perfect matching, we will be able to say that it does with probability greater than half. Therefore this is an RNC algorithm to determine if a perfect matching exists in a bipartite graph G.

1.2.2 General Graph - Decision

We follow similar techniques to determine the existence of a perfect matching in a general graph $G(V, E)$ with $|V| = 2n$ and $|E| = m \leq 4n^2$

Let $A_{2n\times 2n}$ be the adjacency matrix of G such that

$$a_{ij} = \begin{cases} 1 & \text{if } \exists \text{ an edge between vertices } i \text{ and } j \\ 0 & \text{otherwise} \end{cases}$$

Note that unlike the bipartite adjacency matrix, this matrix is always symmetric. We define the Tutte matrix T of G such that

$$t_{ij} = \begin{cases} a_{ij}x_{ij} & \text{if } \ i \leq j \\ -a_{ij}x_{ji} & \text{if } \ i > j \end{cases} \tag{1.6}$$

This matrix T is skew symmetric. The determinant of this matrix is

$$|T| = \sum_{\sigma \in S_{2n}} Sign(\sigma) \prod_{i=1}^{n} t_{i\sigma(i)} \tag{1.7}$$

Unlike the bipartite case, because of the nature of our matrix T, some of the non-zero terms may cancel each other in the expansion of determinant of T. Every even permutation (permutation without any odd cycle) of S_{2n}in the expansion of the determinant corresponds to one or more perfect matchings in graph G. Given an even permutation $\sigma \in S_{2n}$, we can extract a perfect matching M from it by considering the alternate terms t_{ij} from every even cycle of σ. Therefore if a cycle has length greater than two, we can extract two different perfect matchings from it by alternating on it.

We will now show that if a permutation σ contains at least one odd cycle (σ is not an even permutation), then σ gets canceled by some other term in the expansion of determinant of T

Theorem 1.3. *For every $\sigma \in S_{2n}$ containing an odd cycle, $\exists$ $\sigma' \in S_{2n}$ such that in the expansion of determinant of T the monomial corresponding to σ will differ from the monomial corresponding to σ' only in its sign and therefore they will cancel each other.*

Proof. Let σ be the permutation in the expansion of $|T|$ such that σ contains an odd cycle. Let this odd cycle be C. Because our matrix is skew-symmetric, it is easy to see that $\exists\ \sigma' \in S_{2n}$ such that σ' is same as σ everywhere except on the C where it is reversed. That is, we flip the cycle C to get σ' from σ. As we have seen before, the sign of permutation is the $(-1)^k$ where k is the number of inversions in the permutation. It can also be shown that sign of a permutation is $(-1)^{n-l}$ where l is the number of disjoint cycles in the permutation. As both σ and σ' have the same number of cycles, they have the same sign. Again, because of the way we have constructed T, sign of every term composing the cycle C is flipped $(1 \longleftrightarrow -1)$ while flipping the cycle. Therefore the sign of the monomial corresponding to σ will differ from the monomial corresponding to σ' and therefore they will cancel each other. $\square$

Therefore only the even permutations of S_{2n} may be retained during the expansion of $|T|$.

Theorem 1.4. *$|T|$ not identically zero $\Longleftrightarrow$ G has a perfect matching*

Proof. $\Longrightarrow$ if $|T|$ is non-zero, it contains an even permutation as all the odd permutations get canceled. We have already seen how to extract a perfect matching from an even permutation. $\Longleftarrow$ if G has a perfect matching, then the permutation $\sigma \in S_{2n}$ consisting of only 2-cycles of the form $(t_{iM(i)}t_{M(i)i})$ corresponding to the edges of the matching M will not be canceled by any other permutation and will persist in the expansion of $|T|$. Therefore $|T|$ will not be identically zero. $\square$

Now we can use the Schwartz-Zippel theorem to proceed like we did for bipartite graphs.

1.2.3 General Graphs - Search

We will now look at an algorithm on how to output a perfect matching if one exists. For this we need a famous result proved by Mulmuley Vazirani and Vazirani [3]

Lemma 1.5 (Isolation Lemma). *Let $A = \{a_1, a_2, ..., a_n\}$. Let F be a family of subsets of A. Let $w : A \to \{1, 2, ..., 2n\}$ be a weight function which assigns weights independently and at random to elements of A. The weight of a set $S \subset A$, $w(S) = \sum_{a\in S} w(a)$. The weight* *function is said to be isolating if the set in F having the minimum weight is unique. Isolation lemma states that*

$$P(w \textit{ is isolating}) \geq \frac{1}{2} \tag{1.8}$$

Proof. We shall call an element $a_i \in A$, singular with respect to a weight function w, if $\exists\ S_j, S_k \in F$ such that both S_j, S_k have minimum weight in F and $a_i \in S_j$ and $a_i \notin S_k$. We can see that if there is no singular element in A then the minimum weight set in F is unique. We shall now find the probability for the element a_i to be singular.
We first fix the weights of all the elements other than a_i. We define a real number β_i to be the threshold for a_i such that if $w(a_i) \leq \beta_i$ then a_i is contained in some minimum weight set of F. It is easy to see that if $w(a_i) < \beta_i$ then a_i is contained in every minimum weight set of F. In this case the set having minimum weight in F is isolating after giving weight to a_i if the minimum weight set is isolating before giving weight to a_i. (The weight of a set before giving weight to a_i $w(S_j) = \sum_{a_k \in S_j,\ k \neq i} w(a_k)$). If $w(a_i) \geq \beta_i$ then a_i is not contained in any minimum weight set of F. Therefore, a_i is singular if and only if $w(a_i) = \beta_i$
It is crucial to note that the threshold β_i was defined for a_i without taking into consideration the weight of a_i. Therefore, once the threshold for a_i is fixed,

$$P(w(a_i) = \beta_i) \leq \frac{1}{2n}$$

As the weights for a_i are chosen randomly and independently from $\{1, 2, ..., 2n\}$,

$$P(\text{at least one element is isolating}) \leq n \times \frac{1}{2n} = 1/2$$

Therefore the set having the minimum weight in F is unique (weight function is isolating) with probability $\geq \frac{1}{2}$ □

We shall now prove that if the weight function w on the set of edges of the graph G is indeed isolating and if a perfect matching exists, then it is possible to output a perfect matching in NC. For this, we define a modified Tutte matrix T' such that

$$t'_{ij} = \begin{cases} 2^{w(i,j)} & \text{if } i \leq j,\ \exists \text{ an edge}(i,j) \in E \\ -2^{w(i,j)} & \text{if } i > j,\ \exists \text{ an edge}(i,j) \in E \\ 0 & \text{Otherwise} \end{cases} \tag{1.9}$$

Here $w(i,j)$ is the weight of the edge $(i,j) \in E$. All the terms in the expansion of $|T'|$ are of the form $\pm 2^{w(M_1)+w(M_2)}$. M_1, M_2 may or may not be the same matching. We can see that all the odd permutations cancel out because $|T'|$ is just a special case of the Tutte matrix T.

Theorem 1.6. *If the matching M having the minimum weight w(M) is unique, that is, if the weight function is isolating, then the determinant of* $|T'| \neq 0$. *Further the highest power of 2 that divides* $|T'|$ *is* $2w(M)$

Proof. M is the unique minimum weight perfect matching with weight $w(M)$. Then the term corresponding to this matching, that is, σ with only 2-cycles corresponding to the edges of this matching will be $\pm 2^{2w(M)}$. There will be only one such term as M is unique. This term cannot be canceled by any other term or the difference of any other terms as everything else will be sum or difference of higher powers of 2. All the other terms will be of the form $\pm 2^{w(M_1)+w(M_2)}$ where at least one of M_1 or M_2 is not equal to M and have weight at least $w(M)+1$. Therefore $2^{2w(M)+1}$ will divide all the other terms. The expansion of $|T'|$ will be of the form

$$|T'| = \pm 2^{2w(M)} \pm 2^{2w(M)+1}(y) \tag{1.10}$$

where y is an integer strictly greater than 1. Therefore, regardless of what value y takes, $|T'| \neq 0$ and the highest power of 2 that divides $|T'|$ will be $2w(M)$ □

Given that the minimum weight perfect matching is unique, we will now check for each edge $(i,j) \in E$ if it belongs to the minimum weight perfect matching.

Theorem 1.7. *Let B be the modified Tutte matrix of* $G(V,E)$ *with unique minimum weight perfect matching M with weight w(M). Then for every edge* $(i,j) \in E$

$$\frac{|B_{i,j}|}{2^{2w(M)}} 2^{2w(i,j)} \mod 2 \equiv \begin{cases} 0 & \textit{if } (i,j) \notin M \\ 1 & \textit{if } (i,j) \in M \end{cases} \tag{1.11}$$

Where $B_{i,j}$ *is the modified Tutte matrix of the graph* $G_{i,j}$ *which is obtained by removing the* i^{th} *and the* j^{th} *vertices in G along with all the edges incident on these two vertices.*

Proof. It is important to note that the only perfect matchings of $G_{i,j}$ are the perfect matchings of G that contain (i,j) (with the edge (i,j) removed). If M_1 is a perfect matching of G then the corresponding matching in $G_{i,j}$ will be $M_1 - (i,j)$

1. If $(i,j) \notin M$, then the matchings in $G_{i,j}$ will be of the form $M_1 - (i,j)$ where $w(M_1) > w(M)$. As the weights of all the matchings are integers

$$|B_{i,j}| = (2^{w(M)-w(i,j)+w(M)-w(i,j)+1})(y)(2^{2w(i,j)})$$
$$|B_{i,j}| = (2^{2w(M)+1})(y)$$
$$|B_{i,j}| = 2(2^{2w(M)})(y)$$

   Where y is an integer. Therefore

$$\frac{|B_{i,j}|}{2^{2w(M)}} \mod 2 \equiv 0$$

2. If $(i,j) \in M$, then the matchings in $G_{i,j}$ will be of the form $M_1 - (i,j)$ where $w(M_1) \geq w(M)$. Further, the matching $M - (i,j)$ will be contained in $G_{i,j}$ and this will be the unique minimum weight matching of $G_{i,j}$ due to the way $G_{i,j}$ is constructed. Therefore the highest power of 2 that divides $|B_{i,j}|$ will be $2^{2w(M)-2w(i,j)}$. Therefore,

$$|B_{i,j}| = (2^{2w(M)-2w(i,j)})(z)$$

   Where $z \equiv 1 \mod 2$. Therefore

$$\frac{|B_{i,j}|}{2^{2w(M)}} 2^{2w(i,j)} = z \equiv 1 \mod 2$$

   Hence proved

□

We now have an algorithm to output a perfect matching if one exists. We assign weights randomly and independently to all the edges from the set $\{1, 2, ..., 2|E|\}$. We then construct the modified Tutte matrix B for the graph G. We will compute the highest power of 2 that divides $|B|$ and call it $2w(M)$. In parallel, we compute the quantity $k = \frac{|B_{i,j}|}{2^{2w(M)}} 2^{2w(i,j)} \mod 2$ for all the edges $(i,j) \in E$. We output the set $S \subseteq E$ of all the edges for which $k = 1$. We check if S is a perfect matching. S will be a perfect matching if the weight function is isolating. However the weight function will the isolating with the probability $\geq 1/2$. Therefore our algorithm will output a perfect matching with the probability $\geq 1/2$ if one exists. Therefore this algorithm is in RNC. We can compute the determinant of a matrix in $O(\log^2 n)$ time in NC and this is the only critical step in our algorithm. Therefore, the algorithm to find a perfect matching is in RNC and takes $O(\log^2 n)$ time.

Chapter 2

Perfect Matching: Bipartite Graphs

The algorithm in the previous chapter shows that if the weight function is isolating, finding a perfect matching can be done in NC. The only randomized step was to find such a suitable weight function. We would ideally like to construct a set of at most polynomially many weight functions, such that amongst them at least one is isolating. But this remains an open problem. A quasi-NC algorithm to find a perfect matching in a bipartite graph was given by Fenner, Gurjar and Thierauf in 2016 [6]. They gave a way to assign weights to the edges of a graph such that among $O(n^{6\log n})$ weight assignments at least one is isolating. An NC algorithm would have polynomially many processors working in parallel but this quasi-NC algorithm has $O(n^{6\log n})$ processors. Their main theorem is as follows.

Theorem 2.1. *$\exists\ n^{6\log n}$ weight assignments with weights bounded by $O(n^{6\log n})$ such that at least one assignment isolates a minimum weight perfect matching.*

The weight assignment for a graph $G(V, E)$ is given as follows. Let j be a number such that $j = j_0 + j_1 2^{6\log n} + j_2 2^{2*6\log n} + ... + j_k 2^{6k\log n}$ where $k = \lceil \log n \rceil - 1$ and all j_i are $6\log n$ bit long. The j^{th} weight is as follows (done in parallel for all possible j)

$$w^j(e_i) = B^k(2^i \mod j_0) + B^{k-1}(2^i \mod j_1) + ... + B^0(2^i \mod j_k) \tag{2.1}$$

Here $B \geq n^7$

The number of possible j is $n^{6\log n}$ which is equal to the number of processors needed.

We will ensure that at least one among these j assignments is isolating by ensuring that at least one weight assignment gives non zero circulation to all the cycles in the graph G. The circulation of a cycle $C = \{e_1 e_2 ... e_{2m}\}$ with respect to weight w is defined as $c_w(C) = |w(e_1) - w(e_2) + w(e_3) - ... - w(e_{2m})|$. It is important to note that G being bipartite implies that all cycles in G are of even length. For this section, we will assume G to be bipartite unless otherwise stated.

Theorem 2.2. *All cycles in graph G have non-zero circulation $\Rightarrow$ the minimum weight perfect matching is unique.*

Proof. Suppose not

Suppose there are two distinct minimum weight perfect matchings M_1 and M_2. Then their symmetric difference $M_1 \Delta M_2$ contains a cycle C. (The graph $M_1 \Delta M_2$ has edges $E(M_1 \Delta M_2) = E(M_1) \cup E(M_2) - E(M_1 \cap M_2)$). By our assumption, C has non-zero circulation. We can therefore alternate on C to obtain a matching M_3 with weight less than the minimum weight, leading to a contradiction. Therefore M_1 and M_2 are not distinct. Therefore the minimum weight perfect matching is unique. □

If we can somehow ensure that all cycles in G will have non-zero circulation then the minimum weight perfect matching will be unique. One way to do this would be to give weight $w(e_i) = 2^i$ to all the edges. We want the weights to be polynomially bounded even though the number of cycles may be exponential. We will therefore successively consider the graph G_i which is the union of minimum weight perfect matchings of G_{i-1} with $G_0 = G$ in such a way that G_i will have considerably less number of cycles than G_{i-1}. We can show that if the number if cycles is polynomially bounded we can find a weight function with polynomially bounded weights such that all of these cycles have non-zero circulation.

Theorem 2.3. *If G is a graph with n vertices, then for any number s, there is a way for us to construct $O(n^2 s)$ weight assignments with weights bounded by $O(n^2 s)$ such that at least one of the weight assignments gives non-zero circulation to all s cycles for any set of s cycles.*

Proof. As we have seen before, the weight function $w(e_i) = 2^i$ will work but the weights are exponential. We claim that for a given set of s cycles, at least one of the weight functions

$$w_k(e_i) = 2^i \mod k, k \in \{1, 2, ..., t\} \tag{2.2}$$

will assign non-zero circulation to all s cycles if t is large enough. That is, we want to show that $\exists k \leq t$ such that

$$\forall i \leq s, c_{w \mod k}(C_i) \neq 0$$

This will be true provided that $\exists k \leq t$ such that

$$\prod_1^s c_w(C_i) \neq 0 \mod k$$

For this to be true, it is enough to ensure that

$$lcm(2, 3, ..., t) \nmid \prod_1^s c_w(C_i)$$

This will be true if we ensure that $lcm(2, 3, ..., t)$ is greater than $\prod_1^s c_w(C_i)$ which is bounded above by $2^{n^2 s}$. Furthermore we have that $lcm(2, 3, ..., t) \geq 2^t$ for $t \geq 7$. Therefore $t = n^2 s$ is large enough for our purpose. As we are taking the weights modulo k with $k \leq t$ the weights are bound by $t = n^2 s$.

□

This theorem tells us that if s, the number of cycles, is polynomially bound, we can find an acceptable weight function. The motivation to work with successive graphs G_is is that we will eliminate polynomially many cycles every time we go from G_{i-1} to G_i. To do this we will need to show that every cycle which has been given non-zero circulation in G_{i-1} will not have all of its edges present in G_i which is the union of all minimum weight perfect matchings of G_{i-1}. In other words at least one edge of any cycle C which has non-zero circulation in G_{i-1} will be absent in G_i. However, to prove this statement, we need another result about perfect matchings in a bipartite graph.

Theorem 2.4. *Let $G(V, E)$ be a bipartite graph with a weight function w. Let E_1 be the union of all the edges of G which are present in at least one minimum weight perfect matching of G. Then the only perfect matchings of the graph $G(V, E_1)$ are the minimum weight perfect matchings of $G(V, E)$. In other words the weight of all the perfect matchings of $G(V, E_1)$ is the same.*

Proof. As we saw earlier, the description of the perfect matching polytope of a bipartite graph $G(V,E)$ is given by the following conditions. A vector $\overrightarrow{\boldsymbol{x}} = (x_e)_{e\in E} \in \mathbb{R}^m$ belongs to $PM(G)$ if and only if:

$$\sum_{e\in\delta(v)} x_e = 1 \tag{2.3}$$

$$x_e \geq 0 \tag{2.4}$$

A face of this polytope is given by setting some of the inequalities to equalities. In other words a face F of the polytope is characterized by a subset S of E. For the face F we have,

$$x_e = 0, \quad e \in S$$

$$x_e \geq 0, \quad e \in E - S$$

Therefore, if we take $E - S = E_1$ as defined in the statement of the theorem, then we get the face of the polytope spanned by the minimum weight perfect matchings of G. Since all the matchings in this face have equal weight, we get that the only perfect matchings of $G(V, E_1)$ are the minimum weight perfect matchings of G □

Theorem 2.5. *If C is a cycle in the graph $H(V,E_1)$ which is the graph of union of minimum weight perfect matchings of $G(V,E)$, then the circulation of C is zero*

Proof. By our previous theorem, we know that all the perfect matchings of H have equal weight. Therefore the convex hull of the perfect matchings of H equals the convex hull of minimum weight perfect matchings of G. Let $\overrightarrow{\boldsymbol{x}}$ be a vector in the $PM(H)$ such that

$$\overrightarrow{\boldsymbol{x}} = (\overrightarrow{\boldsymbol{x_1}} + \overrightarrow{\boldsymbol{x_2}} + ... + \overrightarrow{\boldsymbol{x_t}})/t \tag{2.5}$$

where $\overrightarrow{\boldsymbol{x_i}}$ are the perfect matching points of $PM(H)$. Let $w(\overrightarrow{\boldsymbol{x}}) = w(\overrightarrow{\boldsymbol{x_i}}) = p$. For $\overrightarrow{\boldsymbol{x}} = (x_e)_{e\in E_1}$ we have $\forall e \in E_1, x_e \neq 0$ as every edge of H features in some minimum weight perfect matching. Further $x_e \geq 1/t$. Let $C = (e_1 e_2 ... e_{2k})$ be a cycle in H with the edges e_i are in order. We define a new vector $\overrightarrow{\boldsymbol{y}}$ such that

$$y_e = \begin{cases} x_e + (-1)^i\epsilon, & \text{if } \ e = e_i \ \text{ for } \ i \in \{1,2,...,2k\} \\ x_e, & \text{otherwise} \end{cases}$$

So we are alternately increasing and decreasing the value of x_e along the cycle C while leaving the value of x_e untouched on the non-C edges. For every vertex v of the cycle C, we increase the weight of one of the edges of C incident on it by ϵ and decrease the weight of one of the edges of C incident on it by ϵ, thus maintaining $\sum_{e\in\delta(v)} y_e = 1$. Also, $y_e \geq 0$ for a sufficiently small ϵ. Therefore $\overrightarrow{\boldsymbol{y}} \in PM(H)$ and $w(\overrightarrow{\boldsymbol{y}}) = p$. We have $w(\overrightarrow{\boldsymbol{x}} - \overrightarrow{\boldsymbol{y}}) = w(\overrightarrow{\boldsymbol{x}}) - w(\overrightarrow{\boldsymbol{y}}) = 0$. But $\overrightarrow{\boldsymbol{x}} - \overrightarrow{\boldsymbol{y}}$ is non-zero only along the edges of C. Therefore $w(\overrightarrow{\boldsymbol{x}} - \overrightarrow{\boldsymbol{y}}) = \epsilon * c_w(C) = 0$. Therefore $c_w(C) = 0$ □

The above theorem guarantees that if the circulation of a cycle C is non-zero in G_{i-1} then at least one of its edges is absent in G_i. So we will proceed by giving non-zero circulation to all cycles of length 4 in G_0 so that at least one edge from all of these 4-cycles will be absent in G_1. We will then give non-zero circulation to all cycles of length 8 in G_1 and construct G_2 and so on. We will double the length of the target cycles every time while going from G_{i-1} to G_i so that we are done in $\lceil \log n \rceil - 1$ stages. We can ensure that we have to give non-zero circulation to at most polynomially many cycles in every stage with the help of following theorem.

Theorem 2.6. *If a bipartite graph G has no cycles of length $\leq r$, then the number of cycles of length $\leq 2r$ is bounded by n^4*

Proof. Let C be a cycle on the vertices $(v_1, v_2, ..., v_l)$ such that $l \leq 2r$. We can successively associate four vertices (u_0, u_1, u_2, u_3) with C such that the distance $(u_0, u_1), (u_1, u_2), (u_2, u_3), (u_3, u_0)$ $l/4$. There is not a unique way to select these four vertices, but we claim that once the vertices are selected, they uniquely determine C.

Suppose not

Suppose there are two cycles of C and C' of length $\leq 2r$ that pass through (u_0, u_1, u_2, u_3). Therefore $\exists(u_i, u_j)$, a pair of successively chosen vertices such that there are two distinct paths between u_i, u_j. As distance between u_i and u_j is $\leq l/4$, it follows that there exists a cycle C'' contained in the walk $(u_i, ..., u_j, ...u_i)$ of length $\leq l/4 + l/4 = l/2 = r$ in the graph G which contradicts our condition on G. Hence (u_0, u_1, u_2, u_3) once chosen, uniquely determine C. The number of ways we can choose these four vertices can be at most $n * (n - 1) * (n - 2) * (n - 3)$ which is $\leq n^4$. Therefore the number of cycles of length at most $2r$ is at most n^4 provided that the graph does not contain any cycle of length $\leq r$ □

We will start by taking $G = G_0$ and then selecting a weight function such that all cycles of length four (the smallest cycles in the graph) have non-zero circulation and are not present in G_1. Then we modify the weight function slightly such that all 8-cycles in G_1 get non-zero circulation so that they are not present in G_2 and so on. We give non-zero circulation to cycles of length 2^{i+2} (polynomially many from last theorem) in G_i so that we are done in $\lceil \log n \rceil$ stages. It is important to note that we do not actually compute G_i in our algorithm, we assign weights (in one go) to the edges such that among $n^{6 \log n}$ weight assignments done in parallel, at least one gives non-zero circulation to all cycles. Recall that our weight function is:

$$w^j(e_i) = B^k(2^i \mod j_0) + B^{k-1}(2^i \mod j_1) + ... + B^0(2^i \mod j_k)$$

Here $k = \lceil \log n \rceil - 1$. The weight function can be thought of as having $\lceil \log n \rceil$ different parts, one part each for cycles of length $2^{i+2}, , 0 \leq i \leq \lceil \log n \rceil - 1$. Each part has n^6 many possibilities for weights, at least one of which gives non-zero circulation to all of the at most n^4 cycles. Our choice of $B = n^7$ ensures that one part of the weight function does not interfere with the other. Given a weight function w^j among $n^{6 \log n}$ functions, we can define a partial function ${w_p}^j$ of w such that

$$ {w_p}^j = \sum_{l=0}^{p} B^{k-l}(2^i \mod j_l) \tag{2.6}$$

Here $p \leq k$ and j, k, B are as defined in the main weight function. Our choice of $B = n^7$ ensures that

$$ {w_p}^j(M_1) \geq {w_p}^j(M_2) \longrightarrow {w_{p+1}}^j(M_1) \geq {w_{p+1}}^j(M_2)$$

This statement implies that only the matchings which have minimum weight with respect to ${w_p}^j$ can also have minimum weight with respect to ${w_{p+1}}^j$. In other words, while going from ${w_p}^j$ to ${w_{p+1}}^j$, we only need to consider the graph of union of minimum weight perfect matchings with respect to ${w_p}^j$. This is what we mean while talking about going from G_i to G_{i+1}.

Once we have assigned weights to the edges, we follow the algorithm given my Mulmuley, Vazirani and Vazirani to output a perfect matching in at least one among the $O(n^{6 \log n})$ instances. Therefore the algorithm takes $O(\log^2 n)$ time to run.

Chapter 3

Pseudo-deterministic Bipartite Perfect Matching

This algorithm uses polynomially many processors working in parallel and poly-log many random bits to give an output in poly-log time. Pseudo-determinism means that the output that we get, if we run the algorithm several times, is the same except in negligible number of cases. A pseudo-deterministic algorithm to find a perfect matching in a bipartite graph was given by Goldwasser and Grossman in 2016 [10]

We will first introduce some techniques to understand this algorithm.

Theorem 3.1. *Let $G_1(V, E_1)$ be the graph of union of minimum weight perfect matchings of bipartite graph $G(V, E)$ with respect to a weight function w. We can construct $G_1(V, E_1)$ in RNC.*

Proof. We will check for every edge e_i if it belongs to some minimum weight perfect matching with respect to w. We will do this in parallel for all edges $e_i \in E$
To check if e_i belongs to some minimum weight perfect matching, we will define

$$w_i(e_j) = \begin{cases} 2e_j & \text{if } i \neq j \\ 2e_j - 1 & \text{if } i = j \end{cases}$$

This weight function w_i implies that e_i is in some minimum weight perfect matching with respect to w if and only if it is in every minimum weight perfect matching with respect to w_i. We have already seen how to find a minimum weight perfect matching in a weighted graph in RNC using the isolation lemma. We therefore compute a minimum weight perfect matching with respect to w_i and check if e_i belongs to that matching. If it does, we say that $e_i \in E_1$. We can therefore compute $G_1(V, E_1)$, the graph of union of minimum weight perfect matchings of bipartite graph $G(V, E)$ in $O(\log^2 n)$ time.

Next, we shall describe some properties of a particular weight function. □

Theorem 3.2. *Let w be a weight function on the ordered set of edges E such that the i^{th} element has the weight*

$$w(i) = p^{2k} i + p^{2(k-1)}(i^2 \mod p) + p^{2(k-2)}(i^3 \mod p) + ... + p^0(i^{k+1} \mod p) \qquad (3.1)$$

Here p is a prime greater than n^2 and $k = \lceil 2 \log n \rceil$. Let the weight of a subset $S \subseteq E$ be $w(S) = \sum_{e \in S} w(e)$
Then no two subsets $A, B \subseteq E$ of size at most k have the same weight.

Proof. Let $A = \{a_1, a_2, ...a_k\}$ and $B = \{b_1, b_2, ...b_k\}$ be subsets of E such that $w(A) = w(B)$. If the sizes of A or B are not exactly k, we can add zeros to the sets to make them exactly k.
Note that $p \geq n^2$ which means that by shifting the powers in the weight function by p^2, we ensure that no coefficient will interact with any other coefficient in the weight function as i can be at most n^2. Even the coefficients of powers of p in the sum

$$w(A) = p^{2k}(a_1 + a_2 + ... + a_k) + ... + (a_1^{k+1} \mod p + a_2^{k+1} \mod p + ... + a_k^{k+1} \mod p)$$

will be kn^2 which is strictly less than p^2 and therefore will not interact with each other.
Therefore $w(A) = w(B)$ implies that all of the following equivalences hold modulo p

$$\begin{gathered} a_1 + a_2 + ... + a_k \equiv b_1 + b_2 + ... + b_k \\ a_1^2 + a_2^2 + ... + a_k^2 \equiv b_1^2 + b_2^2 + ... + b_k^2 \\ ... \\ a_1^{k+1} + a_2^{k+1} + ... + a_k^{k+1} \equiv b_1^{k+1} + b_2^{k+1} + ... + b_k^{k+1} \end{gathered}$$

Newton's identities tell us the equations

$$\begin{aligned}
q_1 &= a_1 + a_2 + ... + a_k \\
q_2 &= a_1^2 + a_2^2 + ... + a_k^2 \\
&... \\
q_k &= a_1^k + a_2^k + ... + a_k^k
\end{aligned}$$

uniquely determine the polynomial of degree k of which a_i are the roots. Therefore they uniquely determine a_i. But this same polynomial also uniquely determines b_i. Therefore we get that for some $\sigma \in S_k$

$$\forall i,\ a_i \equiv b_{\sigma(i)} \mod p$$

But as $a_i, b_i < p, \forall i$, we get that for some $\sigma \in S_k$,

$$\forall i,\ a_i = b_{\sigma(i)}$$

$\therefore A = B$

□

The above theorem ensures that with the weight function w as defined above, no cycles of circulation less than or equal to $\lceil 4 \log n \rceil$ have zero circulation. We shall now prove that each of these cycles will have non-zero circulation for the weight function $w \mod j$ for some $j \in \{2, 3, ...t\}$ for a small enough t

Theorem 3.3. *$\exists\ O(s \log n)$ weight assignments with weights bounded by $O(s \log n)$ such that all cycles of length less than or equal to s have non-zero circulation in at least one of the weight assignments.*

Proof. Let w be as defined in the previous theorem. That is

$$w(i) = p^{2k} i + p^{2(k-1)}(i^2 \mod p) + p^{2(k-2)}(i^3 \mod p) + ... + p^0(i^{k+1} \mod p)$$

Let $w_j = w \mod j$. We need to show that for a cycle C there exists a $j \leq t$ such that $c_{w_j}(C) \neq 0$. From previous theorem, we know that $c_w(C) \neq 0$ $(s = 2k)$. Therefore it is enough to ensure that

$$lcm(2, 3, ..., t) \nmid c_w(C)$$

The right side of this equation is bounded by $2^{O(s \log n)}$. We know that for $t \geq 7, lcm(2, 3, ..., t) \geq 2^t$. Therefore $t = O(s \log n)$ suffices for our purpose. That is for every cycle C of length less than or equal to S $\exists j \in \{2, 3, ...O(s \log n)\}$ such that $c_{w_j}(C) \neq 0$ $\square$

We now have an algorithm to construct the graph G_t from G such that G_t has girth greater than $4 \log n$. We do this by constructing a graph G_j from G_{j-1} (with $G = G_1, j \leq t$) such that G_j is the union of minimum weight perfect matchings of G_{j-1} with respect to the weight function $w_j = w \mod j$. We have already seen that if a cycle has non-zero circulation in a bipartite graph H then it has at least one of its edges absent in the graph of union of minimum weight perfect matchings of H. Our previous algorithm ensures that a cycle of length less than or equal to $4 \log n$ will be eliminated at some point while going from G to G_t where $t = O(\log^2 n)$. Therefore G_t will have girth greater than $4 \log n$. We have already seen how to compute G_j from G_{j-1} in RNC in $O(\log^2 n)$ time. Therefore calculating G_t from G can be done in RNC in $O(\log^4 n)$ time.

We know that if a graph H has girth greater than or equal to $4 \log n$, then it has average degree less that or equal to 2.5. That is, it has at least $\frac{n}{10}$ vertices of degree 2 or less. [11]

So now that we have computed G_t, we have a way to contract it.

It is easy to see that any vertex v of degree 2 or less in a graph H can be contracted along the edges incident on it to get a graph H' such that H' has at least 2 vertices less than H and a perfect matching in H' easily extends to a perfect matching in H. If v has degree 0, then H has no perfect matching as our algorithm terminates. If v has degree 1, then the edge incident on v, e_v has to be in every perfect matching and therefore we can remove that edge along with both the vertices on which it is incident. If v has degree 2, then we can contract v along with both of its neighbors to a single vertex v' and a matching in H' extends to a matching in H. It remains to be seen how we can apply this procedure for several vertices in parallel.

We shall now see how to construct a graph G'_t from G_t such that G'_t has at most $\frac{39}{40}n$ vertices of G_t. We know that G_t has $\frac{n}{10}$ vertices of degree 2 or less. We first check if G_t has any vertices of degree 0. If it does we need proceed no further. We then check of G_t had $\frac{n}{20}$ or more vertices of degree 1. If yes, we can easily shrink all of them in parallel to get our required graph G'_t. If not, then G_t has at least $\frac{n}{20}$ vertices of degree 2. G_t being a bipartite graph, one partition of G_t has at least $\frac{n}{40}$ vertices of degree 2. Let us call the set of these vertices V'. We shall now consider the graph H such that H is the graph consisting of all the vertices in V' along with all the edges adjacent to all of these vertices.

We can construct the connected components of H in NC by computing in parallel for all vertices $v \in V'$ the uv connectivity for all $u \in V'$ (again done in parallel). Let us say that a connected component C of H has l vertices from V'. The number of edges in C will be $2l$ as every vertex in V' has degree exactly 2. If C has less than $2l$ vertices there will not be a perfect matching in C by Hall's theorem. There cannot be more than $2l+1$ vertices in C as it is connected and number of edges is $2l$. Therefore C has $2l$ or $2l+1$ vertices.

If C has $2l$ vertices (if C is an even component), it can have only one cycle because of the degree constraint on the vertices in in V' and because C is connected. Therefore C can have a maximum of two matchings as the symmetric difference of any two distinct matchings always contains a cycle. We know that if the number of matchings in a graph is bounded, we can compute all of them in NC [5]. Note that any perfect matching in G_t restricted to C has to be a perfect matching in C. Therefore a perfect matching in G_t breaks down into a perfect matching in C and a perfect matching in $G_t \backslash C$. Therefore we can find a perfect matching in all the connected even components of H in parallel (in NC) and then remove all of them to get our required graph G'_t.

If C has $2l+1$ vertices (if C is an odd component), it is a tree because it is connected and has $2l$ edges. A tree can have at most one perfect matching. Therefore we can shrink C to a single vertex c and make c adjacent to all the vertices which are adjacent to any vertex in C but are not in C. The contracted graph is still bipartite as the only vertices in C which can have neighbors outside of C are the vertices of C which are not in V'. Therefore we can put c is the partition opposite to V' and all of its neighbors in the same partition as V'. Similar to the previous case, we can shrink all of these odd components in parallel to get our required graph G'_t.

Therefore, we can construct a graph G'_t from G_t such that G'_t has at most $\frac{39}{40}n$ vertices as that of G_t. The critical step here is constructing the connected components which take $O(\log^2 n)$ time

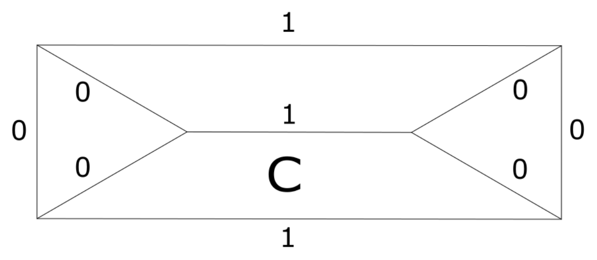
To summarize, we construct the graph G_t from G in RNC in $O(\log^4 n)$ time and then we construct the graph G'_t from G_t in NC in $O(\log^2 n)$. After that take $G'_t = G$ and recurse. As we are reducing a fixed fraction of vertices in every iteration, the number of iterations that we have to perform is $\log n$. Therefore the entire algorithm to construct a perfect matching takes $O(\log^5 n)$ time to run.

We note that randomization is only needed while computing the graph of union of minimum weight perfect matchings. As there can be only one such graph, correctness implies uniqueness.

Chapter 4

Perfect Matching: General Graphs

The problem of searching for a perfect matching in a general graph G is more complicated than the bipartite case. This is mostly due to the fact that giving a cycle non-zero circulation does not ensure that it is not present in the graph of union of minimum weight perfect matchings of G. For example consider the following graph G.



In this graph, the cycle C has been given non-zero circulation. We can see that the graph contains three minimum weight perfect matchings of weight 1. The the graph of union of all the minimum weight perfect matchings of the graph G is G itself. The cycle C is not eliminated as it would have been in the bipartite case. We used the perfect matching polytope of a graph to ensure that C will be eliminated. In the bipartite case, this polytope is given by the following constraints.

$$\textstyle\sum_{e\in\delta(v)} x_e = 1 \quad \forall v \in V \tag{4.1}$$
$$x_e \geq 0 \tag{4.2}$$

However these constraints are not enough in the general case. The perfect matching polytope for general graphs is given by:[12]

$$\sum_{e\in\delta(v)} x_e = 1 \qquad \forall v \in V \tag{4.3}$$
$$x_e \geq 0 \tag{4.4}$$
$$\sum_{e\in\delta(S)} x_e \geq 1 \quad S \subset V, |S| \text{ is odd} \tag{4.5}$$

In the bipartite case, when we say that the cycle C has been eliminated in the graph of union of minimum weight perfect matchings of G, we mean that we have moved from a face of a polytope to one of its subfaces because we have essentially set one of the inequality in $x_e \geq 0$ to an equality. This is what we mean by removing an edge. If we remove enough edges, the subface that we obtain is essentially a perfect matching point. And the number of edges that we can remove is polynomially bounded.

In general graphs however, we have an additional set of inequalities called the odd set constraints. While moving to a subface of a face, even if we do remove an edge, some more odd set constraints might become tight. That is,

$$\sum_{e\in\delta(S)} x_e = 1 \quad S \subset V, |S| \text{ is odd}$$

And the number of such odd sets may not be polynomially bounded. So we need to figure out a way to deal with such constraints. A quasi NC algorithm to find a perfect matching in a general graph was given by Svensson and Tarnowski in 2017 [7]. They dealt with the two main problems regarding generalizing the FGT algorithm for the general case. The first problem is the one we discussed that giving a cycle non-zero circulation does not ensure that it disappears in the graph of union of minimum weight perfect matchings. The second problem is that the graph being a bipartite one was crucial towards proving that if there are no cycles of length up to l, then the number of cycles of length up to $2l$ will be at most n^4 where n is the number of vertices of the graph G. It is the first problem which requires majority of the effort while the second problem remains a technical one. We prove that, if a graph contains no cycles of length upto l, then the number of cycles of length up to $2l$ is at most n^{17} for a particular set of weight functions.

The main problem (departure from the bipartite case) occurs when a cycle crosses a tight odd set,that is when it enters the odd set on an odd numbered vertex in the cycle and leaves it on an even numbered vertex or vice versa. (where all the vertices of a cycle of length $2l$ are ordered from 1 to l). Here our bipartite case strategy to remove the cycle does not work because alternately increasing and decreasing the weights of all the edges in the cycle by a small quantity ϵ will make

$$\sum_{e \in \delta(S)} x_e < 1$$

Where S is the tight odd set crossed by the cycle.
This will violate the constraints on the perfect matching polytope.

Considering that this is the main problem, it is important to note that if in our general graph G, no cycle crossed a tight odd set, our problem of finding a perfect matching would be essentially similar to the bipartite case. That is, we can remove a cycle which does not cross any tight odd set in the same way we would have removed it in the bipartite case by giving it non-zero circulation. Which means that we can apply $\log n$ weight functions one after another to ensure that after this the only cycles which are present in our graph G are the cycles which cross some tight odd set. If at this point, there were no tight odd sets, then we would be done which is not necessarily true in general graphs.

While dealing with tight odd sets, we would like the property that, for any tight odd set S, once we fix an edge $e \in \delta(S)$, it induces a unique perfect matching within S. We shall call such a S for which this property holds, as contractible. It is important to note that if S is a tight odd set, then $V - S$ is also a tight odd set. So if we make all the tight odd sets contractible, then for any tight odd set S and a $e \in \delta(S)$, a matching M in a graph reduces to two subinstances; a unique matching inside S, a unique matching inside $V - S$ and the edge e. The number of ways we can choose such an e is n^2 so all we need to do is ensure different weights to all these matchings. This is equivalent to having a set of n^4 inequalities on the weight function w and it is possible to find a single weight function w satisfying this.

As we mentioned earlier, we will essentially talk in the terms of moving to a subface of a face of the perfect matching polytope to isolate a perfect matching. A face F in the polytope for general graphs is completely characterized by two things; a subset $A \subseteq E$ such that $x_e = 0, \forall\ e \in A$ and a family $S(F)$ of tight odd sets. An important property of the perfect matching polytope is that the family $S(F)$ is completely characterized by a maximal laminar family Λ of $S(F)$. A family Λ of sets is said to be laminar if for any two sets $A, B \subseteq \Lambda$, either $A \cap B = \phi$ or $A \subseteq B$ or $B \subseteq A$. It has also been shown that any maximal laminar family of subsets of $S(F)$ is enough to describe a face. When we move to the subface of a face, we do not need to consider an entirely new maximal laminar family, we can just add a few sets to our existing family to make it a maximal laminar family for that subface. If, after applying a series of weight functions according to the bipartite case, we get a laminar family describing a face to be a chain ($S_1 \subset S_2 \subset S_3 ...$), then we see that there is no cycle contained within any single layer and so that case is easy to solve. However, we have no reason to assume that our family will be a chain.

Our strategy will be to make all sets of size less than four contractible while giving non-zero circulation to all cycles of length up to four, then make all sets of size less than eight contractible while e giving non-zero circulation to all cycles of length upto eight and so on. We shall call this parameter as $l - goodness$. We call a face laminar pair (F, Λ) to be l good for some parameter l if Λ is a maximal laminar subset of $S(F)$ and

1. All subsets of Λ of size up to l are F contractible.

2. In (F, Λ, l) contraction of the graph G, there are no cycles of node weight up to l.

We contract contractible sets the same way that we shrink blossoms in edmonds algorithm [1], that is, contract the sets to a single vertex v and $\delta(v) = \delta(S)$. In order to remove cycles from (F, Λ, l) contraction of the graph G, we want the certain cycles to not respect the face F. For this, we will use a series of weight functions for $t \geq 7$

$$w_k(e_j) = (4n^2 + 1)^j \mod k, k \in \{2, 3, ..., t\} \tag{4.6}$$

Where $t = O(n^{20})$. We call $F[w]$ the subface of F minimized with respect to w. So instead of minimizing with respect to one weight function at a time we will concatenate the weight function such that $F[w \circ w'] = F[w][w']$ where

$$w \circ w'(e) = n^{21} w(e) + w'(e) \tag{4.7}$$

In order to remove cycles in the general case, it is not enough to just give non-zero circulation to all even cycles, we also have to give non-zero circulation to all even walks because the symmetric difference of even cycles may also contain odd cycles. We will define a vector $(\pm 1)_C$ on a walk C such that $(\pm 1)_C$ has value 1 on even numbered edges of C, -1 on the odd numbered edges of C and zero elsewhere. We say that C respects a face if

1. edges of C belong to the edges of $F[w]$
2. $\langle (\pm 1)_C, 1_{\delta(S)} \rangle = 0$ for $S \in S(F[w])$

Two of the important results of this paper is that;

Theorem 4.1. *If a alternating walk C has been given non-zero circulation in F with respect to w, then it will not respect the face F[w].*

Theorem 4.2. *We can find a set of $O(n^3 s)$ weight functions with polynomially bounded weights such that for any s-set of alternating walks, we can give non-zero circulation to all s walks.*

Now that we have these two theorems, we will think about contracting sets in faces. We recall that we want all tight odd sets to be contractible, that is, once we fix an edge coming out of that set, the entire matching within the set is uniquely determined. However instead of making all the tight odd sets in a face F contractible, we will make some sets contractible and then move to a subface of F. We are helped in this course by a result:

Theorem 4.3. *If S is a set tight for a face F, if F' is a subface of F and if S is F contractible, then S is F' contractible.*

For any face laminar pair (F, Λ), where Λ is a maximal laminar family of sets tight for F, we have that (F, Λ) is always 1-good, because single vertices are trivially contractible and there is no cycle of node- weight up to 1. In fact, (F, Λ) is 2-good. We will successively try to get a face laminar pair (F', Λ') which is $2l$-good from a face laminar pair (F, Λ) which is

l-good where F' is a subface of F and $\Lambda \subseteq \Lambda'$. If $l \geq n$, where n is the number of vertices in the graph, then $|F| = 1$ and were are done. Now we come to one of the most important theorems in this paper

Theorem 4.4. *There exists a weight function w among $(\log^2 n + \log n)$ weight functions such that PM[w] = 1*

Here w is obtained by concatenation of $\log n$ weight function, concatenated in the way as defined before in 4.7. Therefore we need an $(n^{21 \log^2 n + \log n})$ weight functions to guarantee a weight function which is isolating. The number of times we have to move from l-good to $2l$-good is $\log n$ and therefore the algorithm runs in poly-log time using $(n^{21 \log^2 n + \log n})$ many processors.

Chapter 5

Popular Matchings

5.1 Characterization

Popular matchings can be defined on graphs whose vertices have preferences regarding who to be matched to in a matching. These preferences may be with or without ties. A vertex v is said to prefer a matching M over M' if either it is matched in M but unmatched in M' or if it matched to a vertex w in M and w' in M' with w being strictly higher than w' on v's preference list. v is said to indifferent between M and M' if it is unmatched in both or matched in both to vertices which tie on v's preference list. A matching M is said to be popular if for all matchings M', the number of vertices that prefer M over M' is greater than or equal to the number of vertices that prefer M' over M.

A polynomial time algorithm to find a popular matching in a bipartite graph where only one partition has preferences regarding who to be matched to, was given by Abhraham, Irving, Telikepalli and Mehlhorn in 2007 [8].

Throughout this chapter, we consider the bipartite graph $G(A \cup P, E)$ where vertices $a \in A$ (also known as applicants) have preferences about vertices $p \in P$ (also known as posts). A brute force method to find a popular matching is to compare all the matchings in a graph G. But the number of matchings in a graph may be exponential. We therefore need to find a characterization for popular matchings.

We shall first introduce some conventions which we will use throughout. G is always a bipartite graph unless stated otherwise. We will denote $a_i \rightarrow \{p_{k_1}, (p_{k_2}, p_{k_3}), p_{k_4}, ..., p_{k_j}\}$ as the preference list of applicant a_i. The round brackets indicates that p_{k_2} and p_{k_3} are ranked the same in the preference list of a_i. It is important to note that j need not be equal to $|P|$, that is, the preference list for a_i need not be a complete preference list for the all posts in P a_i will only give preferences between those vertices to which it is connected in G. We shall denote E_j to be the set of rank j edges such $\forall\ a_i, E_j$ consists of edges from a_i to all the posts which are ranked j^{th} on the preference list of a_i. It is important to note that because of ties, a_i can have multiple edges of rank j incident on it. We can now take the first step towards characterizing popular matchings. We add a last resort post l_i unique to each a_i, which, as its name suggests is last without any ties on $a_i^{\prime}s$ preference list. We do this so that it is always possible to find an applicant complete matching in $G(A \cup P, E)$. Therefore the preference list of a_i is $a_i \rightarrow \{p_{k_1}, (p_{k_2}, p_{k_3}), p_{k_4}, ..., p_{k_j}, l_i\}$. We are now ready to begin to characterize popular matchings.

Theorem 5.1. *A matching M is a popular matching of $G(A \cup P, E)$ if M restricted to $G_1 = G(A \cup P, E_1)$ is a maximum matching of G_1.*

Proof. We shall call M restricted to G_1 as M_1. Suppose that M_1 is not a maximum matching of G_1. Then there exists an M_1 augmenting path Q in G_1. Let this path be $(a_1 p_1 ... a_k p_k)$ where both a_1 and p_k are free vertices in G_1 with respect to the matching M_1. Because we have added last resort posts, a_1 has to be matched by M in G. But p_k can be either matched or remain unmatched by M.

- Suppose that p_k is unmatched by M in G. Then we can alternate along the augmenting path Q to get a new matching M' which is more popular than M. That is, all the non-matched edges of Q become the matched edges in M' and matched edges in Q become unmatched in M'. Everything else remains the same. This is possible p_k is free in M. Due to the nature of the graph G_1, a_1 prefers M' over M but all the other applicants in the path Q are indifferent between M' and M. Therefore M' is more popular than M. Therefore M is not a popular matching.

- Suppose that p_k is matched by M in G to $M(p_k)$. Clearly p_k is not a first ranked post in $M(p_k)$ as it is unmatched by M_1. To construct M', we again alternate along Q ie assign a_1 to p_1 ... a_k to p_k. We assign $M(p_k)$ to any of its first ranked posts say p_{k+1}. At worst p_{k+1} would already have been matched in M and so $M(p_{k+1})$ will become free in M' and we therefore match it to its last resort post. But a_1 and $M(p_k)$ will both prefer M' over M and $M(p_{k+1})$ may prefer M over M'. In any case M' is more popular than M. Therefore M is not a popular matching.

Therefore M restricted to G_1 is a maximum matching of G_1. The notations M_1 and G_1 will mean the same thing throughout this chapter. □

Next, we shall divide all of the vertices of the graph $G(A \cup P, E)$ into three categories, $EVEN, ODD, UR$. Consider a maximum matching M_1 of G_1. Then M_1 partitions the vertices of G into three categories $EVEN(M_1), ODD(M_1), UR(M_1)$. The set $EVEN(M_1)$ consists of all those vertices v_i such that there is an M_1 alternating path of even length between v_i and any free vertex with respect to M_1 in G_1. The set $ODD(M_1)$ consists of vertices v_i such that there is an M_1 alternating path of odd length between v_i and any free vertex with respect to M_1 in G_1. The set $UR(M_1)$ consists of vertices v_i that are unreachable from all free vertices with respect to M_1.

Theorem 5.2 (Gallai-Edmonds)**.** *[13] This theorem states that:*

1. *the sets $EVEN(M_1), ODD(M_1), UR(M_1)$ are pairwise disjoint and they are the same for any maximum matching M_1 of G_1 (they are a property of the graph and not the maximum matching). Therefore we can divide the vertices of graph G into three sets. $EVEN$, ODD, UR*

2. *In any maximum cardinality matching (of G_1), every vertex in ODD is matched to to some vertex in $EVEN$ and every vertex in UR is matched to another vertex in UR.*

3. *In any maximum cardinality matching no vertex in EVEN can be matched to another vertex in EVEN. Therefore the size of a maximum cardinality matching is $|ODD| + \frac{UR}{2}$*

4. *There can be no edge in G_1 between a node in UR and a node in $EVEN$*

We also consider two important subsets for posts in the set P in $G(A \cup P, E)$. We call P_1 the set of first ranked posts. That is $p_i \in P_1$ if p_i is top preference post for some applicant

$a \in A$. We call P_2 the set of second ranked posts. That is $p_i \in P_2$ if p_i a top ranked even post for some applicant $a \in A$. (a post $p \in P_2$ if $p \in EVEN$ and there exists $a \in A$ such that all posts p' which are higher on $a's$ preference list than p do not belong to the set $EVEN$). It is important to note that $P_1 \cap P_2$ need not be empty. $P_1(a)$ is the set of first ranked posts of a (all the top preference posts). $P_2(a)$ are the top ranked posts in $a's$ preference list which belong to the set $EVEN$. $P_1(a) \cap P_2(a)$ can also be non empty.

We now come to an important characterization for popular matchings.

Theorem 5.3. *M is a popular matching for the graph $G(A \cup P, E) \implies$ all applicants $a \in A$ are either matched to a post in $P_1(a)$ or in $P_2(a)$*

Proof. Suppose not

Suppose an applicant a is matched by M to a post $M(a)$ which ranks strictly higher than $P_2(a)$ but strictly lower than $P_1(a)$ in the preference list of a. Therefore $M(a)$ has to be an odd or unreachable post in G_1. But $M(a)$ is not a first ranked post of a. Therefore $M(a)$ has to be unmatched in G_1. But all odd or unreachable posts have to be matched in any maximum matching of G_1. Therefore M restricted to G_1 is not a maximum matching. Therefore M is not a popular matching which is a contradiction.

Suppose an applicant a is matched by M to a post $M(a)$ which ranks strictly lower than $P_2(a)$ on the preference list of a. Let p be any post in $P_2(a)$. If p is unmatched by M, then we can promote a to p to get a more popular matching, leading to a contradiction. Therefore suppose M matches p to an applicant a'. Then there can be two cases:

- Suppose $p \notin P_1(a')$. Then we can promote a to p, a' to some post $p' \in P_1(a')$ and the applicant $M(p')$ to its last resort post, to obtain a matching M' more popular than M. Therefore M is not a popular matching.

- Suppose $p \in P_1(a')$. As $p \in P_2(a)$, p is an even post. Therefore there exists an even length alternating path Q' in G_1 from $p = p_1$ to a vertex p_k which is free in G_1. Here Note that p_k might still be matched by M. Note that a is free in G_1. Therefore we can define an M_1 augmenting path Q in G_1 such that $Q = aQ' = (ap...p_k)$. Therefore we can define a new matching M' such that the non-matching edges of Q become matched in M' and the matched edges become unmatched. We also promote $M(p_k)$ (if it exists) to its first preference post say p_{k+1} and and $M(p_{k+1})$ to its last resort post. Therefore $a, M(p_k)$ prefer M' to M while $M(p_{k+1})$ prefers M to M'. (If $M(p_k)$ does not exist then a prefers M' to M while no applicant prefers M to M') Therefore M' is more popular than M which is a contradiction.

Therefore p cannot be strictly worse than $P_2(a)$
Therefore $p \in P_1(a) \cup P_2(a)$ □

We are now ready to give a complete characterization for popular matchings.

Theorem 5.4. *M is a popular matching of the graph $G(A \cup P, E) \Longleftrightarrow$*

1. *M restricted to G_1 is a maximum matching of G_1.*
2. *$\forall\ a \in A, M(a) \in P_1(a) \cup P_2(a)$*

Proof. $\Longrightarrow$ We have already proved this in the last two theorems

$\Longleftarrow$ Suppose not. Suppose there is a matching M' more popular than M. We are going to show that for every applicant a that prefers M' to M, there is a unique applicant a' that prefers M to M'. Let a be an applicant that prefers M' to M. So $M(a) \in P_2(a)$, otherwise it never would have preferred M'. Therefore $M'(a)$ has to be an odd or a unreachable post. Let $M \oplus M'$ be the symmetric difference of M and M'. Then $M \oplus M'$ restricted to E_1 consists of disjoint paths and cycles. We are going to show that $M'(a)$ is always contained in a non-empty path of $(M \oplus M') \cap E_1$. a is not matched by M in G_1 ie $(a, M(a)) \notin E_1$ as $M(a) \in P_2(a)$. Therefore a is definitely not contained in a cycle. Now $M'(a)$ being an odd or unreachable vertex is contained in every maximum matching of G_1. Therefore it is contained in a non-empty path or cycle of $(M \oplus M') \cap E_1$. But $M'(a)$ is matched to a in M' and a is not contained in any cycle. Therefore $M'(a)$ is contained in a non-empty path in $(M \oplus M') \cap E_1$ with one endpoint as either a or $M'(a)$. It is a if $M'(a) \in P_1(a)$, it is $M'(a)$ otherwise (In which case a is isolated).

So for every applicant a that prefers M' to M, there is a distinct non-empty path Q in $(M \oplus M') \cap E_1$. Since $M'(a)$ is an odd or unreachable post, it follows that every post in Q must be odd or unreachable as well and has to be matched in every maximum matching. So the other end of Q must be an applicant a' which prefers M to M' since Q is alternating. Therefore for every applicant a that prefers M' to M, there is a unique applicant a' that prefers M to M'

Therefore M' is not more popular than M. Therefore M is a popular matching.

□

We now have a characterization for popular matchings. We shall now see an algorithm to find a popular matching.

5.2 The Algorithm

1. Given a graph $G(A \cup P, E)$, find a maximum matching M_1 in the graph G_1.
2. With the help of M_1, partition the vertices of G into three sets $EVEN$, ODD, UR.
3. Compute $P_1(a), P_2(a)$ for every vertex a to find the sets P_1 and P_2.
4. Find the induced subgraph G' of G on the vertex set $A \cup P_1 \cup P_2$
5. In G' remove all the edges connecting two nodes in ODD or a node in ODD to a node in UR.
6. Find a maximum matching M in G' by augmenting M_1.
7. If M is not applicant complete, return that 'No popular matching exists. Otherwise, return M.

This algorithm is pretty straightforward. Step 5 is necessary so as to ensure that after successively augmenting M_1, any maximum matching we obtain is a maximum matching when restricted to G_1. Therefore in a popular matching M, no two nodes in ODD or a node in ODD and a node in UR can have an edge between them. Also by the fourth condition of Gallai-Edmonds theorem, there can be no edge between an unreachable and a even node in G_1. Therefore unreachable nodes will only be matched to other unreachable nodes and as all of them are matched in M_1, all of them will be matched in M. Because we obtain M by successive augmentation M has $|ODD| + \frac{|UR|}{2}$ edges of rank 1. Therefore M is a maximum matching when restricted to G_1. Therefore M is a popular matching.

The crucial time consuming steps in the algorithm are steps 1 and 6 both of which will take $O(\sqrt{n}m)$ time by the algorithm given by Hopcraft-Karp. Therefore the running time of the algorithm is $O(\sqrt{n}m)$.

We can find a maximum cardinality popular matching using a similar algorithm. We find a popular matching M in G' by following the previous algorithm. Then from M and G' we remove all the edges of the form $(a, l(a))$ where $l(a)$ is the last resort post of a to get a matching M'. Note that M' is still a maximum matching of G_1 as no edges of the form $(a, l(a))$ are of rank 1. We then successively augment M' to get a matching N which will be our maximum cardinality popular matching.

5.3 Strict Preferences

If we consider a simplified version of the same popular matching problem when all the applicants a have strict preferences between the posts that they are matched to, we can find a popular matching in linear time. Here for an applicant a, the only post in $P_1(a)$ is the post at the top of $a's$ preference list and the only post in $P_2(a)$ is the post highest in the preference list of a that is not a first preference post for any other applicant (This has degree 0 in G_1 and therefore even). Therefore $P_1 \cap P_2 = \phi$ As the first preference post is unique, we get that every post in P_1 has to be matched. Further, it has to be matched to an applicant for which it is a first preference post as a popular matching M restricted to G_1 is maximum matching.

In G' all the applicants have degree 2. Also, the number of edges is twice the number of applicants in G'. To find an applicant complete matching, we first successively remove all degree 1 posts in G' along with the edges incident on them (and their endpoints) as these have to be contained in any applicant complete matching. We put these edges in our required matching M'. We remove all isolated posts after this process is complete. All of this takes linear time. G' then splits into various connected components. Then for any connected component C we can use Hall's theorem to check if an applicant complete matching exists. Such an applicant complete matching has to exist for every component C. All applicants in any component will still have degree two and therefore the number of edges will be twice as that of the applicants in that component. As the average degree is 2, and no vertices of degree 1 in C, all vertices will have degree exactly 2 and will be a cycle. We therefore take the alternate edges in this cycle for our required matching M' which is applicant complete. This too takes linear time. If any $p \in P_1$ is unmatched in this matching M', we match p to any applicant for which it is a first preference post to get our required popular matching M. We can do this in linear time.

Chapter 6

Conclusions

Abhraham, Irving, Telikepalli and Mehlhorn gave a novel way to characterize popular matchings. The problem of finding an NC algorithm for computing a popular matching remains an open one.

In case of perfect matchings, a major open problem is finding a perfect matching using polynomially many processors working in parallel. It is not clear if we can do this by extending the techniques introduced by Fenner, Gurjar and Thierauf. This is because we need $\log n$ iterations to eliminate all the desirable cycles. Some new insights will be needed to circumvent this to get an NC algorithm to find a perfect matching.

Bibliography

[1] J. Edmonds, “Paths, trees, and flowers,” *Canadian Journal of mathematics*, vol. 17, no. 3, pp. 449–467, 1965.

[2] J. E. Hopcroft and R. M. Karp, “An nˆ5/2 algorithm for maximum matchings in bipartite graphs,” *SIAM Journal on computing*, vol. 2, no. 4, pp. 225–231, 1973.

[3] K. Mulmuley, U. V. Vazirani, and V. V. Vazirani, “Matching is as easy as matrix inversion,” in *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pp. 345–354, ACM, 1987.

[4] D. Y. Grigoriev and M. Karpinski, “The matching problem for bipartite graphs with polynomially bounded permanents is in nc,” in *Foundations of Computer Science, 1987., 28th Annual Symposium on*, pp. 166–172, IEEE, 1987.

[5] M. Agrawal, T. M. Hoang, and T. Thierauf, “The polynomially bounded perfect matching problem is in nc 2,” in *Annual Symposium on Theoretical Aspects of Computer Science*, pp. 489–499, Springer, 2007.

[6] S. Fenner, R. Gurjar, and T. Thierauf, “Bipartite perfect matching is in quasi-nc,” in *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pp. 754–763, ACM, 2016.

[7] O. Svensson and J. Tarnawski, “The matching problem in general graphs is in quasi-nc,” in *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 696–707, Oct 2017.

[8] D. J. Abraham, R. W. Irving, T. Kavitha, and K. Mehlhorn, “Popular matchings,” *SIAM Journal on Computing*, vol. 37, no. 4, pp. 1030–1045, 2007.

[9] S. J. Berkowitz, “On computing the determinant in small parallel time using a small number of processors,” *Information processing letters*, vol. 18, no. 3, pp. 147–150, 1984.

[10] S. Goldwasser and O. Grossman, “Perfect bipartite matching in pseudo-deterministic rnc.,” in *Electronic Colloquium on Computational Complexity (ECCC)*, vol. 22, p. 208, 2015.

[11] N. Alon, S. Hoory, and N. Linial, “The moore bound for irregular graphs,” *Graphs and Combinatorics*, vol. 18, no. 1, pp. 53–57, 2002.

[12] J. Edmonds, “Maximum matching and a polyhedron with 0, 1-vertices,” *Journal of research of the National Bureau of Standards B*, vol. 69, no. 125-130, pp. 55–56, 1965.

[13] L. Lov, “z, md plummer, matching theory,” *Annals of discrete mathematics*, vol. 29, 1986.