# Report: Polyhedral Combinatorics, Matroids and Derandomization of Isolation Lemma

*Instructor: Rohit Gurjar*

Soham Chatterjee

sohamchatterjee999@gmail.com
Website: sohamch08.github.io

2024

# Contents

Chapter 1

# Perfect Matching Polytope

## 1.1 Matching Polytope

## 1.2 Perfect Matching Polytope

**Definition 1.2.1: Perfect Matching Polytope**

Let $G = (V, E)$ be a graph. For any perfect matching $M$ of $G$, consider the incidence vector $x^M = (x_e)_{e \in E} \in \mathbb{R}^E$ given by

$$c_e^M = \begin{cases} 1 & \text{if } e \in M \\ 0 & \text{o/w} \end{cases}$$

For any perfect matching $M$ of $G$ this vector $x^M$ is called as a *Perfect Matching Point*. The bipartite perfect matching polytope of the graph $G$ is defined to the convex hull of all its perfect matching points,

$$PM(G) = \mathrm{Conv}\{x^M \mid M \text{ is a perfect matching in } G\}$$

## 1.3 Bipartite Perfect Matching Polytope

It also defined like the perfect matching polytope where we just take the graph to be a bipartite graph. The following lemma form [LP86] gives a simple description of the perfect matching polytope of a bipartite graph $G$

**Theorem 1.3.1** [LP86]
Let $G = (V, E)$ be a bipartite graph and $x = (x_e)_{e \in E} \in \mathbb{R}^E$. Then $x \in PM(G)$ if and only if

$$\sum_{e \in \delta(v)} x_e = 1 \quad v \in V,$$
$$x_e \geq 0 \quad e \in E$$

where for any $v \in V$, $\delta(v)$ denotes the set of edges incident on the vertex $v$.

CHAPTER 2

# Bipartite Perfect Matching

## 2.1 A RNC Algorithm for Search-PM

We will recall the RNC algorithm of Mulmuley, Vazirani and Vazirani [MVV87] for the construction of a perfect matching (Search-PM). Though the algorithm works for any graph, we will only consider the bipartite graphs here.

Let $G = (V, E)$ be a bipartite graph with vertex partitions $L = \{u_1, \ldots, u_{\frac{n}{2}}\}$ and $R = \{v_1, \ldots, v_{\frac{n}{2}}\}$ and a weight function $w$. Consider the following $\frac{n}{2} \times \frac{n}{2}$ matrix $A$ associated with $G$,

$$A(i,j) = \begin{cases} 2^{w(e)} & \text{if } e = (u_i, u_j) \in E \\ 0 & \text{otherwise} \end{cases}$$

The algorithm then computes the determinant of $A$. Now

$$\begin{aligned} \det(A) &= \sum_{\pi \in S_{\frac{n}{2}}} sgn(\pi) \prod_{i=1}^{\frac{n}{2}} A(i, \pi(i)) \\ &= \sum_{M\text{:PM in } G} sgn(M) 2^{w(M)} \end{aligned}$$

The second equation holds since the product $\prod_{i=1}^{\frac{n}{2}} A(i, \pi(i))$ is nonzero if and only if the permuation $\pi$ corresponds to a perfect matching. Here $sgn(M)$ is the sign of the corresponding permutation.

If the graph $G$ does not have a perfect matching then we have $\det(A) = 0$. However if the graph $G$ has perfect matchings, $\det(A)$ maight equal to 0 due to cancellation due to $sgn(M)$. To avoid such cancellations, one needs to design the weight function $w$ cleverly. In particular if $G$ has a perfect matching and $w$ is isolating then the minimum weight perfect matching can not be canceled with other terms as there is unique minimum weight perfect matching.

Given an isolating weight assignment for $G$, one can construct the minimum weight perfect matching in NC. Let $M^*$ be the unique minimum weight perfect matching in $G$ w.r.t $w$. First we find $w(M^*)$ be finding out the highest power of 2 that divides $\det(A)$ since after $2^{w(M^*)}$ will not divide the monomial corresponding to $M^*$ and that monomial survives in $\det(A)$. So more that after $2^{w(M^*)}$ it will not divide $\det(A)$. Now choose an $(u_i, u_j) = e \in E$ then compute the determinant of the minor $A_{i,j}$ which is basically associated with $G - e$. If the highest power of 2 that divides $\det(A_{i,j})$ is larger than the $2^{w(M^*)}$ then $e \in M^*$. Doing this in parallel for each edge, we can find all the edges in $M^*$.

By Theorem ?? we have an isolating weight with high probability. Moreover the weights chosen by the isolation lemma are polynomially bounded. Therefore the entries of in matrix $A$ have polynomially many bits. Hence it suffices to compute the determinant in $\text{NC}^2$ [Ber84]. Also the construction is in $\text{NC}^2$. There fore this yields an RNC-algorithm for Search-PM.

## 2.2 A Quasi-NC Algorithm using Isolation

Let $G = (V, E)$ be given bipartite graph. In the following discussion we will assume that $G$ has perfect matchings. Our goal is to isolate one of the perfect matchings in $G$ by any appropriate weight function. We will also show that if $G$ does not have any perfect matchings then our algorithm will detect this. In the following discussion we will prove this theorem

**Theorem 2.2.1** [FGT16, Theorem 3.1]
For bipartite graphs, PM and SEARCH-PM are in QUASI-NC$^2$

We will construct an isolating weight function for bipartite graphs. The idea is to create a weight function which ensures nonzero circulations for a small set of cycles in a black-box way i.e. without having being able to compute the set efficiently. Then we will show that if we construct a smaller graph wrt this weight function then we don't have those small cycles with nonzero circulations then we have the number of cycles with twice the size of the previous ones are polynomially bounded. Then we proceed to create a new weight function which will give nonzero circulations to all the cycles with twice the size. And this way we will continue. This same type of idea we will repeatedly use with necessary modifications in chapter 4 and chapter 6.

The idea above to create a weight function which gives nonzero circulation to every nice cycles in $G$ actually works because then we have unique perfect matching by Lemma **??**

### 2.2.1 Isolating Small Cycles

The following lemma describes a standard trick to create a weight function for a small set of cycles in graph.

**Lemma 2.2.2** [CRS93]
Let $G$ be a graph with $n$ vertices. Then for any number $s$, one can construct a set of $O(n^2 s)$ weight assignments with weights bounded by $O(n^2 s)$, such that for any set of $s$ cycles, one of the weight assignments gives nonzero circulation to each of the $s$ cycles.

***Proof:*** Let us first assign exponentially large weights. Let $e_1, e_2, \ldots, e_m$ be some enumeration of the edges of $G$. Define a weight function $w$ by $w(e_i) = 2^{i-1}$ for $i \in [m]$. Then clearly every cycle has a nonzero circulation. However we want to achieve this with small weights.

We consider the weight assignment modulo small numbers i.e. the weight function is $\{w \bmod j \mid 2 \le j \le t\}$ for some appropriately chosen $t$. We want to show that for any fixed set of $s$ cycles $\{C_1, \ldots, C_s\}$ one of these assignments will work when $t$ is chosen large enough.

Now we want

$$\exists\, j \le t,\ \forall\, i \le s,\ c_w(S_i) \neq 0 \iff \exists\, j \le t,\ \prod_{i=1}^{s} c_w(C_i) \neq 0 \bmod j$$

In other words we want

$$lcm(2,3,\ldots,t) \nmid \prod_{i=1}^{t} c_w(C_i)$$

Hence if we take $t$ such that $lcm(2,3,\ldots,t) > \prod_{i=1}^{t} c_w(C_i)$ then we are done.

Now the product $\prod_{i=1}^{t} c_w(C_i)$ is bounded by $2^{n^2 s}$. This is because with exponential weights like in the RNC algorithm of section 2.1 we have an isolating perfect matching so we need weights less than that and therefore the new weights are bounded by the exponential weights for which weight of a cycle can at most be $2^{n^2}$ and since there are $s$ many cycle we have the bound $2^{n^2 s}$. So if we have $t$ such that $lcm(2,3,\ldots,t) > 2^{n^2 s}$ then we are done. Now $lcm(2,3,\ldots,t) > 2^t$ for $t \ge 7$. Thus choosing $t = n^2 s$ suffices. Clearly the weights are bounded by $t = n^2 s$. ■

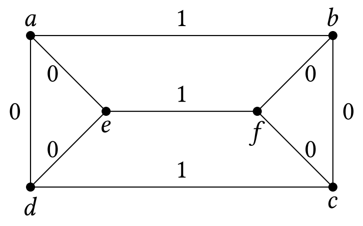
### 2.2.2 Union of Minimum Weight Perfect Matchings

Let us assign a weight function for bipartite graph $G$ which gives nonzero circulations to all small cycles. Consider a new graph $G_1$ which obtained by the union of minimum weight perfect matchings in $G$. Out hope is that $G_1$ is significantly smaller than $G$.

**Note:-**

We don't know if $G_1$ can be efficiently created from $G$ as determinant of the bi-adjacency matrix with weights in the like in the RNC algorithm of section 2.1 be zero and therefore we can not use that way to obtain perfect matchings. We will show we don't need to construct $G_1$. It is just used n the argument. Our final weight assignment will be completely black-box

We will also show by the following lemma that why this technique only works for biparitte graphs, not in general graphs i.e. $G_1$ constructed from minimum weight perfect matchings in $G$ contains no other prefect matching than these. For general graph this does not hold:



In this graph we will denote the edge connected vertices $a, b$ to be $e_{ab}$. And this way we will denote all the edges. Then the minimum weight perfect matchings have weight 1 and they are

$$\{e_{ad}, e_{bc}, e_{ef}\}, \quad \{e_{ac}, e_{bf}, e_{cd}\}, \quad \{e_{de}, e_{cf}, e_{ab}\}$$

Then their union has the perfect matching $\{e_{ab}, e_{cd}, e_{ef}\}$ which has weight 3 and not a minimum weight perfect matching.

The fact that $G_1$ has only minimum weight perfect matching is equivalent to saying that every nice cycle has zero circulation. The following lemma proves even stronger statement that every cycles has zero circulation (not necessarily nice cycles.)

**Lemma 2.1** [FGT16, Lemma 3.2]

Let $G = (V, E)$ be a bipartite graph with weight function $w$. et $C$ be a cycle in $G$ such that $c_w(C) \neq 0$. Let $E_1$ be the union of all minimum weight perfect matchings in $G$. Then the graph $G_1 = (V, E_1)$ does not contain $C$.

***Proof:*** Consider the perfect matching polytope of $G$, $PM(G)$. Let the weight of the minimum weight perfect matching in $G$ be $q$. Let $x_1, x_2, \dots, x_t$ be all the minimum weight perfect matching points of $G$ i.e. corners of $PM(G)$ corresponding to weight $q$. Consider the average point $x \in PM(G)$ of these perfect matching points, $x = \frac{x_1+x_2+\cdots+x_t}{t}$. Clearly we have $w(x) = q$. And since each edge of $G_1$ participates in a minimum weight perfect matching for $x = (x_e)_{e\in E}$ we have $x_e \neq 0$ $\forall\ e \in E$.

Now consider a cycle $C$ with $c_w(C) \neq 0$. Suppose $C = (e_1, e_2, \dots, e_k)$ and all the edges of $C$ are in $E_1$. We will show that if we move from point $x$ along the cycle $C$ we reach a point in $PM(G)$ with a weight smaller than $q$.

Consider the point $y$ defined as

$$\forall\ e \in E, \quad y_e = \begin{cases} x_e + (-1)^i\epsilon \& \text{if } e = e_i \text{ for some } i \in [k] \\ x_e & \text{o/w} \end{cases}$$

for some $\epsilon \neq 0$. Clearly $x - y$ has nonzero coordinates only on the edges of the cycle $C$, by alternating between $\epsilon$ and $-\epsilon$. Hence

$$w(x) - w(y) = w(x - y) = \pm c_w(C) \neq 0$$

Now we take $\epsilon$ in the following way:

- Take $sgn(\epsilon)$ such that $w(x - y) > 0$.
- Take $\epsilon$ small enough such that $y_e \geq 0\ \forall\ e \in E$.

After choosing such $\epsilon$ since $w(x) - w(y) = w(x - y) > 0$ we have $q = w(x) > w(y)$. Now we will show that $y \in PM(G)$. To show that we will sow that $y$ fulfills the conditions of Theorem 1.3.1. Now the second condition that $y_e \geq 0$ for all $e \in E$ is already satisfied by the choice of $\epsilon$. So we only need to show that for any $v \in V$

$$\sum_{e\in\delta(v)} y_e = 1$$

To show this we consider 2 cases:

**Case 1:** $v \notin C$. Then $\forall\ e \in \delta(v)$ we have $e \notin C$. So $y_e = x_e$. Since $x \in PM(G)$ we have

$$\sum_{e\in\delta(v)} x_e = 1 \implies \sum_{e\in\delta(v)} y_e = 1$$

**Case 2:** $v \in C$. Let $e_j$ and $e_{j+1}$ are the edges incident on $v$ in $C$. Then

$$y_{e_j} = x_{e_j} + (-1)^j\epsilon \qquad \text{and} \qquad y_{e_{j+1}} = x_{e_{j+1}} + (-1)^{j+1}\epsilon \qquad \forall\, e \in \delta(v) - \{e_j, e_{j+1}\},\ y_e = x_e$$

So

$$\begin{aligned}\sum_{e\in\delta(v)} y_e &= \left[\sum_{e\in\delta(v)-\{e_j,e_{j+1}\}} x_e\right] + \left[x_{e_j} + (-1)^j\epsilon\right] + \left[x_{e_{j+1}} + (-1)^{j+1}\epsilon\right] \\ &= \left[\sum_{e\in\delta(v)} x_e\right] + (-1)^j\epsilon + (-1)^{j+1}\epsilon = \sum_{e\in\delta(v)} x_e = 1\end{aligned}$$

So the point $y$ satisfies the property $\sum\limits_{e\in\delta(v)} = 1$ forall $v \in V$. Hence $y \in PM(G)$. Now since $w(y) < q$ there must be a corner point of the polytope which corresponds to a perfect matching in $G$ with weight less than $q$. This contradicts the minimality of $q$. Hence $C$ is not in $G_1$. ■

This technique of moving along the cycle has been used by Mahajan and Varadarajan in [MV00]. Now We will show another proof of this lemma by Rao, Shpilka and Wigderson in [GG17].

***Alternate Proof*** *[GG17, Proof of Lemma 6]:* Let $G'$ be the multigraph obtained by taking disjoint union of all minimum weight perfect matchings (i.e. if an edge appears in $k$ many minimum weight perfect matchings of $G$ then $G'$ contains $k$ copies of the edge.

**Note:-**
$G'$ is a regular graph since it is a disjoint union of matchings and matchings are regular graph of degree 1.

Suppose there exists a cycle $C$ of non zero circulation in $G_1$. Since the cycle is in $G_1$ then this cycle is also in $G'$. WLOG assume that the sum of the weights of the odd edges of $C$ is larger than the sum of the weights of the even edges. Then we can remove a single copy of each odd edges of $C$ from $G'$ and add a single copy of each even edges of $C$ to $G'$ and we call this new graph $G''$

Suppose $q$ be the minimum weight of a matching in $G$. Suppose $G$ has $d$ minimum weight matchings. Then sum of the weights of the edges in $G'$ is $qd$. However, the total weight of all edges in in $G''$ is lower than the total weight of all edges in $G'$. We know that $G''$ is a regular bipartite graph of degree $d$ and therefore by Lemma **??** it is an union of $d$ perfect matchings.

If we decompose $G''$ into $d$ perfect matchings, it is impossible that they all have weight at least $q$ as $G''$ has total weight less than $qd$. Therefore $G''$ has a matching of weight less than $q$, which contradicts the minimality of $q$. ■

A consequence of this lemma is that $G_1$ has no other perfect matchings than the ones used to define $G_1$ cause if $M_0$ and $M_1$ be two different perfect matchings in $G_1$ then $M_0 \triangle M_1$ forms a set of nice cycles and by the Lemma 2.1 the circulations all of these cycles are 0 and therefore $M_0$ and $M_1$ have same weight and hence they both are minimum weight perfect matchings.

**Corollary 2.2.3**
Let $G = (V, E)$ be a bipartite graph with weight function $w$. Let $E_1$ be the union of all minimum weight perfect matchings in $G$. Then every perfect matching in the graph $G_1 = (V, E_1)$ has the same weight, the minimum weight of any perfect matching in $G$.

### 2.2.3 Bounding Number of Cycles with Length Twice Large of The Smallest Cycle

By our weight function in Lemma 2.2.2 each small cycles in $G$ has a nonzero circulation. Hence, by Lemma 2.1 $G_1$ has no small cycles. Now we want to repeat this procedure with $G_1$ with a new weight function. $G_1$ has no small cycles. Hence, we look at slightly larger cycles (twice larger) and we will argue that their number remains polynomially bounded.

Teo and Koe in [TK92] showed that the number of the shortest cycles in a graph with $m$ edges is bounded by $m^2$. In te following lemma we extend their argument and give a bound on the number of cycles that have length at most twice the length of the shortest cycles.

**Lemma 2.2.4** [FGT16, Lemma 3.4]

Let $G = (V, E)$ be a graph with $n$ nodes that has no cycles of length $\leq r$. Let $r' = 2r$ if $r$ is even and $r' = 2r - 2$ otherwise. Then $H$ has $\leq n^4$ cycles of length $\leq r'$.

***Proof:*** Let

$$C = v_0 \xrightarrow{e_0} v_1 \xrightarrow{e_1} \cdots \xrightarrow{e_{l-2}} v_{l-1} \xrightarrow{e_{l-1}} v_1$$

be a cycle of length $l \leq r'$ in $G$. Now we successively choose 4 nodes in $C$ $(u_0, u_1, u_2, u_3)$ where

$$u_i = v_{\lfloor \frac{il}{4} \rfloor} \quad \forall\, i \in \{0, 1, 2, 3\}$$

Now observe the distance between two successive nodes is $\leq \lfloor \frac{l}{4} \rfloor \leq \frac{r}{2}$. and therefore distance between the nodes $u_3$ and $u_0$ is also $\leq \lfloor \frac{l}{4} \rfloor \leq \frac{r}{2}$.

Since any node of $C$ can be chosen as a starting point $u_0$, the four nodes $(u_0, u_1, u_2, u_3)$ associated with $C$ are not uniquely defined. But by the following claim we will show they uniquely define $C$.

***Claim:*** Cycle $C$ is the only cycle in $G$ of length $\leq r'$ that is associated with $(u_0, u_1, u_2, u_3)$.
***Proof:*** Suppose $C' \neq C$ be a cycle of length $\leq r'$ such that both $C$ and $C'$ are associated with same $(u_0, u_1, u_2, u_3)$. Consider the paths between two successive nodes in both $C$ and $C'$. Since $C \neq C'$ there exists a path $p$ and $p'$ following $C$ and $C'$ respectively connecting two same successive nodes in both $C$ and $C'$ such that $p \neq p'$. Now $p$ and $p'$ form a cycle in $H$ of length

$$|p| + |p'| \leq \frac{r}{2} + \frac{r}{2} = r$$

which is not possible as there are no cycles of length $\leq r$ in $G$. Hence, contradiction. ■

Hence by the claim each tuple of 4 nodes uniquely defines a cycle $C$ in $H$. There are $\leq n^4$ ways to choose 4 nodes and their order. Hence the number of cycles of length $\leq r'$ is at most $n^4$. ■

Lemma 2.2.4 suggests that we continue form $G_1$ and in each successive round, we double the length of cycles and adapt the weight function to give nonzero circulations to these slightly longer cycles (twice larger). By Lemma 2.1 these cycles will disappear from the new graph $G_2$ obtained by taking only minimum weight perfect matching from $G_1$. This way in $\log n$ rounds we reach a graph with no cycles i.e. with a unique perfect matching. In the following section we show how to construct the weight assignment.

### 2.2.4 Constructing Weight Assignment

Let $G = (V, E) = G_0$ be bipartite graph with $n$ nodes that has a perfect mathcing. Define $k = \lfloor \log n \rfloor - 1$ which is the number of successive rounds we will need. Note that the shortest cycles in $G$ have length 4. Then suppose we have defined subgraphs $w_i$ and $G_i$ for $0 \leq i \leq k$ when define

$G_{i+1}$ : The union of minimum weight perfect matchings in $G_i$ according to weight $w_i$

$w_{i+1}$ : A weight function on $G_{i+1}$ such that all cycles in $G_{i+1}$ of length $2^{(i+1)+2}$ have nonzero circulations by Lemma 2.2.2

By the definition of $G_i$ any two perfect matchings in $G_i$ have the same weight, not only according to $w_i$ but also $w_j$ for all $j < i$ for any $i \in [k]$. By Lemma 2.1 graph $G_i$ does not have any cycles of length $\leq 2^{i+1}$ for each $i \in [k]$. In particular $G_k$ does not have any cycles since $2^{k+1} \geq n$. Therefore, $G_k$ has a unique perfect matching.

Our final weight function $w$ will be a combination of $w_0, \ldots, w_{k-1}$. We combine them in a way that the weight assignment in a later round does not interfere with the order of perfect matchings given by earlier round weights. Let $B$ be a number greater thatn the weight of any edge under any of these weight assignments. Then define

$$w = w_0 B^{k-1} + w_1 B^{k-2} + \cdots + w_{k-1} B^0$$

In the definition of $w$, the precedence decreases from $w_0$ to $w_{k-1}$ since once two perfect matchings have same minimum weight with restpect to $w_i$, $w_i$ doesn't participate in the calculations for $w_j$ in $G_j$ with $j > i$. Therefore, for any two perfect matchings $M_1$ and $M_2$ in $G_0$ we have $w(M_1) < w(M_2)$ if and only if there exists an $0 \leq i \leq k - 1$ such that

$$w_j(M_1) = w_j(M_2), \text{ for } j < i, \quad w_i(M_1) < w_i(M_2)$$

As a consequence, the prefect matchings left in $G_i$ have a strictly smaller weight with respect to $w$ than the ones in $G_{i-1}$ that did not make it to $G_i$.

**Lemma 2.2.5** [FGT16, Lemma 3.5]

For any $i \in [k]$ let $M_1$ be a perfect matching in $G_i$ and $M_2$ be a perfect matching in $G_{i-1}$ which is not in $G_i$. Then $w(M_1) < w(M_2)$

***Proof:*** Since $M_1$ and $M_2$ are prefect matchings in $G_{i-1}$ we have $w_j(M_1) = w_j(M_2)$ for all $j < i-1$. From the definition of $G_i$ and Corollary 2.2.3 we have $w_{i-1}(M_1) < w_{i-1}(M_2)$. Hence we have $w(M_1) < w(M_2)$. ∎

Therefore by the lemma it follows that the unique perfect matching in $G_k$ has strictly smaller weight with respect to $w$ than all other perfect matchings.

**Corollary 2.2.6**

The weight assignment $w$ defined

$$w = w_0 B^{k-1} + w_1 + B^{k-2} + \cdots + w_{k-1} B^0$$

is isolating for $G_0$.

Now all it remains is to bound the values of the weight assigned. First we will look at the number of the cycles which need to be assigned a nonzero circulations in each round. So in first round, we give nonzero circulations to every cycle of length 4. Clearly the number of such cycles is $\le n^4$. By Lemma 2.2.2 there are a set of $O(n^6)$ weight assignments with weights bounded by $O(n^6)$ as $s \le n^4$. So $G_1$ does not have any cycles of length 4. In $i$-th round we have the graph $G_i$ that does not have any cycles of length $\le 2^{i+1}$. Now by Lemma 2.2.4 the number of cycles of length $\le 2^{i+2}$ is bounded by $n^4$. So by Lemma 2.2.2 we give a set of $O(n^6)$ weight assignments with weights bounded by $O(n^6)$ whcih gives nonzero circulations to all $\le n^4$ cycles of length $\le 2^{i+2}$. Now the length of the largest cycle can be at most $n$. Hence we only have to iterate like this $O(\log n)$ times to have all cycles.

Recall that the number $B$ used in defining $w$ is the highest weight assigned by any $w_i$. Hence $B = O(n^6)$ as for all each round each set of weight assignments have their weights bounded by $O(n^6)$. Therefore the weihts in the assignment $w$ are bounded by $B^k = O(n^{6 \log n})$. Or we can say the weights have $O(\log^2 n)$ bits.

For each $w_i$ there are at most $O(n^6)$ possibilities. We don't know which one would work. Therefore we try all of them and take all possible combinations to create $w$ and then we try all of them. In total we need to try $O(n^{6 \log n})$ weight assignments. This can be done in parallel. Hence clearly every weight assignment can be constructed in QUASI-NC$^1$. Therefore we proved:

**Theorem 2.2.7** [FGT16, Lemma 3.7]

In QUASI-NC$^1$ one can construct a set of $O(n^{6 \log n})$ integer weight functions on $\left[\frac{n}{2}\right] \times \left[\frac{n}{2}\right]$ where the weights have $O(\log^2 n)$ bits, such that for any given bipartite graphs with $n$ nodes, one of the weight functions is isolating.

With this construction of weight assignments, we can decide the existsence of a perfect matching in a bipartite graph in QUASI-NC$^2$ as follows: We take the biadjacency matrix $A$ from section 2.1 ehich has entry $2^{w(e)}$ for edge $e$. We compute $\det(A)$ for each of the constructed weight funcions in parallel. If the given graph has a perfect matching, then one of the weight functions isolates a perfect matching. As we discussed in section 2.1 for this weight function $\det(A)$ will be nonzero. When there is no perfect matching, then $\det(A)$ will be zero for any weight function.

As our weight function have $O(\log^2 n)$ bits, the determinant entries have quasi-polynomial bits. The deteminant can still be computed in parallel with circuits of quasi polynomial of size $2^{O(\log^2 n)}$ by the algorithm of Berkowitz [Ber84]. As we need to compute $2^{O(\log^2 n)}$-many determinants in parallel, our algorithm is in QUASI-NC$^2$ with circuit size $2^{O(\log^2 n)}$.

To construct a perfect matching we follow the algorithm of Mulmuley in [MVV87] from section 2.1 with each of our weight functions. For a weight function $w$ which is isolating, the algorithm outputs the unique minimum weight perfect matching $M$. If we have a weight function $w'$ which is not isolating, still $\det(A)$ might be non-zero with respect to $w'$ then the algorithm computes a set of edges $M'$ that might or might not be a perfect matching. However, it is easy to verify if $M'$ is indeed a perfect matching and in this case, we will output $M'$. As the algorithm involves computation of similar determinants as in the decision algorithm, it is in Quas-NC$^2$ with circuit size $2^{O(\log^2 n)}$. This finishes the proof of the Theorem 2.2.1.

## 2.3 RNC Algorithm with Fewer Random Bits

We can also present the bipartite matching algorithm in section 2.2 in an alternate way i.e. instead of Quasi-NC we wil get an RNC circuit but with only oly-logarithmically many, namely $O(\log^2 n)$ random bits.

**Note:-**

For complete derandomization, it would suffice to bring the number of random bits down to $O(\log n)$. Then there are only polynomially many random strings which can all be tested in NC. Hence this method is only a log-factor away from derandomization

### 2.3.1 Decision Version

There are two reasons that we need quasi-polynomially large circuits:

(i) We need to try quasi-polynomially many different weight assignments.

(ii) Each weight assignment has quasi-polynomially large weights

For the first problem we modify the Lemma 2.2.2 to get a random weight assignment which works with high probability.

**Lemma 2.3.1** [CRS93, KS01]

Let $G$ be a graph with $n$ nodes and $s \geq 1$. There is a random weight assignment $w$ which uses $O(\log ns)$ random bits and assigns weights bounded by $O(n^3 s \log ns)$ i.e. with $O(\log ns)$ bits such that for any set of $s$, cycles $w$ gives nonzero circulation to each of the $s$ cycles with probability at least $1 - \frac{1}{n}$

***Proof:*** We will follow the process of Lemma 2.2.2 and give exponential weights modulo small numbers. Here we will use prime numbers as moduli. Recall that the weight function $w$ defined by $w(e_i) = 2^{i-1}$. Let us choose a random prime $p$ among the first $t$ primes. We take our random weight assignment to be $w \bmod p$. We want to show that with high probability this weight function gives nonzero circulation to every cycle in $\{C_1, \ldots, C_s\}$ In other words

$$\prod_{i=1}^{s} c_w(C_i) \not\equiv 0 \bmod p$$

as the product is bounded by $2^{n^2 s}$ it has at most $n^2 s$ prime factors. Let us choose $t = n^3 s$. This would mean that a random prime works with probability at least $\left(1 - \frac{1}{n}\right)$. As the $t$−th prime can only be as large as $2t \log t$, the weights are bounded by $2t \log t = O(n^3 s \log ns)$ and hense $O(\log ns)$ bits. A prime with $O(\log ns)$ bits can be constructed using $O(\log ns)$ random bits (see [KS01]) ■

We will do the same as in subsection 2.2.4. However we use the random scheme Lemma 2.3.1 to choose each of the weight functions $w_0, w_1, \ldots, w_{k-1}$ independently. The probability that all of them provide nonzero circulations on their respective set of cycles $\geq \left(1 - \frac{1}{n}\right)^k \geq 1 - \frac{k}{n} \geq 1 - \frac{\log n}{n}$ using the union bound.

Now instead of combining them to form a single weight assignment like in subsection 2.2.4 we use a different variable for each weight assignment. We modify the construction of matrix $A$ from section 2.1. Let $L = \{u_1, \ldots, u_{\frac{n}{2}}\}$ and $R = \{v_1, \ldots, v_{\frac{n}{2}}\}$ be the vertex partition of $G$. For variables $x_0, x_1, \ldots, x_{k-1}$ define an $\frac{n}{2} \times \frac{n}{2}$ matrix $A$ by

$$A(i,j) = \begin{cases} \prod_{i=0}^{k-1} x_i^{w_i(e)} & \text{if } e = (u_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases}$$

And therefore we have

$$\det(A) = \sum_{M:\text{pm in } G} sgn(M) \prod_{i=0}^{k-1} x_i^{w_i(M)}$$

where $sgn(M)$ is the sign of the corresponding permutation. From the construction of weight assignments it follows that if a graph has a perfect matching then lexicographically minimum term in $\det(A)$ with respect to the exponents of variables $x_0, \ldots, x_{k-1}$ in this precedence order, comes from a unique perfect matching. Therefore we have the following lemma:

**Lemma 2.3.2**

$\det(A) \neq 0 \iff G$ has a perfect matching

Since each $w_i$ needs to give nonzero circulations to $n^4$ cycles the weights obtained by the scheme of Lemma 2.3.1 will be bounded by $O(n^7 \log n)$. This means the weight of a matching will be bounded by $O(n^8 \log n)$. Hence $\det(A)$ is a polynomial of individual degree $O(n^8 \log n)$ with $\log n)$ variables. To test if $\det(A)$ is nonzero one can apply the standard randomized polynomial identity test

**Theorem 2.3.3** Schwartz-Zippel Lemma: [Sch80, Zip79]

Let $P(x_1, \ldots, x_n)$ be a nonzero polynomial of $n$ variables with degree $d$ over field $\mathbb{F}$. Let $S$ be a finite subset of $\mathbb{F}$ with at least $d$ elements in it. Then we have:

$$\underset{\alpha_1,\ldots,\alpha_n \in S}{Pr}[p(\alpha_1, \alpha_2, \ldots, \alpha_n) = 0] \leq \frac{d}{|S|}$$

Hence if we plug in random values for variables $x_i$ independently from $\{1, 2, \ldots, n^9\}$ and if $\det(A) \neq 0$ then the evaluation is nonzero with high probability.

**Number of Random Bits:** For a weight assignment $w_i$ we need $O(\log ns)$ random bits from Lemma 2.3.1 where $s = n^4$ by Lemma 2.2.4. Thud the number of random bits required for all $w_i's$ togather is $O(k \log n) = O(\log^2 n)$. Finally, we need to plug in $O(\log n)$ random bits for each $x_i$. This again requires $O(\log^2 n)$ random bits, as discussed above.

**Complexity:** The weight construction involves taking exponential modulo small primes by Lemma 2.3.1. Primality testing can be done by brute force algorithm in $\mathrm{NC}^2$, as the numbers involved have $O(\log n)$ bits. Thus the weight assignments can be constructed in $NC^2$. Moreover, the determinant with polynomially bounded entries can be computed in $\mathrm{NC}^2$ [Ber84].

In summery we get the following theorem,

**Theorem 2.3.4** [FGT16, Theorem 4.1]

For bipartite graphs, there is an $\mathrm{RNC}^2$-algorithm for PM which uses $O(\log^2 n)$ random bits.

### 2.3.2 Search Version

Here we get a similar algorithm for SEARCH-PM using also only $O(\log^2 n)$ random bits. This result in [FGT16] improves the RNC algorithm of Goldwasser and Grossman [GG17] based on an earlier version of [FGT16] that uses $O(\log^4 n)$ random bits. Their RNC algorithm has an additional property that it is *pseudo-deterministic* i.e. it outputs the same perfect matching for almost all choice of random bits. The following algorithm does not have that property.

**Theorem 2.3.5**

For bipartite graphs, there is an $\mathrm{RNC}^3$ algorithm for SEARCH-PM which uses $O(\log^2 n)$ random bits

***Proof:*** Suppose again $G = (V, E)$ be a bipartite graph with vertex partitions $L = \{u_1, \ldots, u_{\frac{n}{2}}\}$ and $R = \{v_1, \ldots, v_{\frac{n}{2}}\}$. Now we will have the weight functions $w_0, \ldots, w_{k-1}$ same as in subsection 2.3.1 by the Lemma 2.3.1. In this case we will combine the weight functions like in subsection 2.2.4 to obtain $w$. Let $M^*$ be the unique minimum weight perfect matching in $G$ with respect to the combined weight assignment $w$.

Now in subsection 2.2.4 of the QUASI-NC algorithm we had a sequence of graph $G_1, G_2, \ldots, G_k$ with $G = G_0$ where we took $G_{i+1}$ to be the graph constructed by taking only the minimum weight perfect matchings in $G_i$ with respect to $w_i$. Now instead of computing $G_1, \ldots, G_k$ in $O(\log^2 n)$ random bits (which is not clear anyway) we will compute a sequence of graphs $H_1, \ldots, H_k$ where $H_i$ will be a subgraph of $G_i$ for each $i \in [k]$. And also each $H_i$ will contain $M^*$ (obviously since we have to find $M^*$). Hence, once we have $H_k = G_k$ we are done.

So we start with $H_0 = G_0 = G$ and let $0 \leq i < k$. We will describe the $i$th round. So suppose we have constructed $H_i = (V, E_i)$. Now we want to compute $H_{i+1}$. Now an edge of $E_i$ will appear in $H_{i+1}$ only if it participates in a matching

$M$ with $w_i(M) = w_i(M^*)$. Therefore, $H_{i+1}$ is a subgraph of $G_{i+1}$ as $H_i$ is a subgraph of $G_{i+1}$. Now for an edge $e$ denote the product $\mathbb{X}_i^{w(e)}$:

$$\mathbb{X}_i^{w(e)} = \prod_{j=0}^{k-1-i} x_{i+j}^{w_i(e)}$$

For a matching $M$ the term $\mathbb{X}_i^{w(M)}$ is defined similarly. Now let $N(e)$ denote the set of edges which are neighbors of an edge in $G_i$ i.e. all edges $e' \neq e$ such that $e' \cap e \neq \emptyset$. Also for an edge $e \in E_i$, define the $\frac{n}{2} \times \frac{n}{2}$ matrix $A_e$ as:

$$A_e(k,l) = \begin{cases} \mathbb{X}_i^{w(e')} & \text{If } e' = (u_k, v_l) \in E_i - N(e) \\ 0 & \text{otherwise} \end{cases}$$

Now the matrix has 0 entry for each neighboring edge of $e$. Thus its determinant is a sum over all perfect matchings which contain $e$. That is

$$\det(A_e) = \sum_{M\text{:PM in } H_i} sgn(M)\, \mathbb{X}_{i+1}^{w(M)}$$

Consider the coefficient $c_e$ of $x_i^{w_i(M^*)}$ in $\det(A_e)$,

$$c_e = \sum_{\substack{M\text{:PM in } H_i \\ w_i(M)=w_i(M^*), e\in M}} sgn(M)\, \mathbb{X}_{i+1}^{w_i(M)}$$

Now define $H_{i+1}$ to be the union of all the edges $e$ for which $c_e \neq 0$.
***Claim:*** $M^* \subseteq E_{i+1}$ i.e. all edges of $M^*$ appears in $H_{i+1}$.
***Proof:*** For any edge $e \in M^*$, the polynomial $c_e$ will contain the term $\mathbb{X}_{i+1}^{w_i(M^*)}$. As the matching $M^*$ is isolated in $H_i$ with respect to the weight vector $\{w_{i+1}, \dots, w_{k-1}\}$ the polynomial $c_e$ is nonzero. ■

For construction of $H_{i+1}$ we need to test if $c_e$ is nonzero for each edge $e \in E_i$. As argued above in the decision part (in subsection 2.3.1) the degree of $c_e$ is $O(n^7 \log^2 n)$. We apply the standard identity testing Theorem ?? and we plug in random values for the variables $x_{i+1}, \dots, x_{k-1}$ independently from $[n^{11}]$. The probability that the evaluation will be nonzero is at least $1 - O\left(\frac{\log^2 n}{n^3}\right)$.

To compute the evaluation, we plug in values of $x_{i+1}, \dots, x_{k-1}$ in $\det(A_e)$ in $\det(A_e)$ and find the coefficient of $x_i^{w_i(M^*)}$. This can be done in $\mathrm{NC}^2$ by [BCP83, Corollary 4.4]. For all the edges we use the same random values for the variables $x_{i+1}, \dots, x_{k-1}$ in each identity test. The probability that e test works successfully for each edge is at least $1 - O\left(\frac{\log^2 n}{n}\right)$ by the union bound. We continue this for $k$ rounds to find $H_i$, which is a perfect matching.

We need again $O(\log^2 n)$ random bits for the weight assignments $w_0, \dots, w_{k-1}$ and the values for the $x_i's$. Note that we use the same random bits for $x_i$ in all $k$ rounds. This decreases the success probability, which is now at least $1 - O\left(\frac{\log^3 n}{n}\right)$ by the union bound.

In $\mathrm{NC}^2$ we can construct the weight asssignments and compute the determinants in each round. As we have $k = O(\log n)$ rounds, the overall complexity becomes $\mathrm{NC}^3$.

■

Chapter 3

# Matroid Polytope

Chapter 4

# Linear Matroid Intersection

# Chapter 5 Matroid Matching

# Chapter 6

# Fractional Matroid Matching

Fractional Matroid Matchings generalizes the case for Matroid Matching or Matroid Parity problem with allowing fractional solutions for the polytope which we will show below. We start with the same kind of state like Matroid Parity Problem

## 6.1 Fractional Matroid Matchings Polytope

Let $M = (E, \mathcal{I})$ is a matroid with ground set $E$ of even cardinality and with elements $E$ is partitioned into lines or pairs. Let $L$ is the set of lines. Let $r : \mathcal{P}(E) \longrightarrow \mathbb{Z}$ be the rank function and $sp : \mathcal{P}(E) \longrightarrow \mathcal{P}(E)$ be the span function. Assume that $\forall\, l \in L, r(L) = 2$. With this setting (same as matroid parity problem) we now define the polytope following [VV92]

**Definition 6.1.1: Fractional Matroid Matching Polytope**

Let $\mathscr{L}$ denote the lattice of flats in $M$ wtih $S_1 \wedge S_2 = S_1 \cap S_2$ and $S_1 \vee S_2 = sp(S_1 \cup S_2)$ and for each line $l \in L$ let $a_l : \mathscr{L} \longrightarrow \{0, 1, 2\}$ be the function $a_l(S) = r(sp(l) \cap S)$. Now for any $S \in \mathscr{L}$ and $x \in \mathbb{R}_+^{|L|}$ let $a(S) \cdot x$ denote the vector $(a(S) \cdot x)_l = a_l(S)x_l$ for any $l \in L$. Then the set

$$FP(M) = \{x \in \mathbb{R}_+^{|L|} \mid : a(S) \cdot x \leq r(S) \text{ for each } S \in \mathscr{L}\}$$

is fractional matroid matching polytope for $M$ and each vector $x \in FP(M)$ is called a fractional matroid matching.

We take $|L| = m$ to imply that originally the ground set has $2m$ elements. Now we can also allow $x$ to be from $\mathbb{R}^m$, not restricting only to positive vectors. This polytope is a subset of $[0, 1]^m$. We will explain the setting wtih the following example:

**Example 6.1**

Consider the matroid $M$ with ground set

$$E = \{a_1, a_2, b_1, b_2, c_1, c_2, d_1, d_2\}$$

where every 4 element subset of $E$ is a base except these 4 sets

$$\{a_1, a_2, b_1, b_2\}, \qquad \{a_1, a_2, c_1, c_2\}, \qquad \{a_1, a_2, d_1, d_2\},$$
$$\{b_1, b_2, c_1, c_2\}, \qquad \{b_1, b_2, d_1, d_2\}, \qquad \{c_1, c_2, d_1, d_2\}$$

Now the lines are defined to be

$$l_1 = \{a_1, a_2\} \qquad l_2 = \{b_1, b_2\}, \qquad l_3 = \{c_1, c_2\}, \qquad l_4 = \{d_1, d_2\}$$

Now the flats of $M$ are empty set, individual elements, every pair of elements, set consists of one element from each of three lines, pair of line and $E$. Hence $FP(M)$ is the set of $x \in \mathbb{R}_+^{|L|}$ satisfying

$$\begin{array}{ccc} 2x_1 + 2x_2 \leq 3 & 2x_1 + 2x_3 \leq 3 & 2x_1 + 2x_4 \leq 3 \\ 2x_2 + 2x_3 \leq 3 & 2x_2 + 2x_4 \leq 3 & 2x_3 + 2x_4 \leq 3 \end{array}$$

$$2x_1 + 2x_2 + 2x_3 + 2x_4 \leq 4$$

$$2x_i \leq 2 \quad \text{for each } i \in [4]$$

Now the we show the theorem Theorem 6.1.1 which states that the fractional matroid matching polytope arises as a linear relaxation of the matroid matching problem.

**Theorem 6.1.1** [VV92, Theoerm 2.1]
An integer vector $x \in \mathbb{R}_+^m$ is the incidence vector of a matroid matching iff $x$ is a fractional matroid matching.

You can clearly see this theorem by comparing the Matroid Matching Polytope and Fractional Matroid Matching Polytope so we are omitting the proof.

**Theorem 6.1.2** [GP13, Theorem 1]
The vertices of the fractional matroid matching are half-integral

### 6.1.1 Weighted Fractional Matroid Matching

**Definition 6.1.2: Weighted Fractional Matroid Matching Problem**

It is to find a fractional matroid matching $x$ that maximizes $w \cdot x$ for a non-negative weight assignment $w : L \longrightarrow \mathbb{Z}_+$.
For plain Fractional Linear Matroid Matching Problem we need to find a fractional matroid matching $x$ which maximizes the size i.e. $L_1$ norm of $x$ which is $\sum\limits_{l \in L} |x_l|$.

Gijswijt and Pap in [GP13] gave a polynomial time algorithm for weighted fractional linear matroid matching. They also gave the following characterization for maximizing face of the polytope with respect to a weight function.

**Theorem 6.1.3** [GP13, Prood of Theorem 1]
Let $L = \{l_1, \ldots, l_m\}$ be a set of lines with $l_i \subseteq \mathbb{F}^n$ and $w : L \longrightarrow \mathbb{Z}$ be a weight assignment on $L$. Let $F$ denote the set of fractional linear matroid matchings maximizing and $S \subseteq [m]$ such that every $x \in F$ has $y_e = 0$ for all $e \in S$. Then for some $k \leq n$, $\exists$ a $k \times m$ matrix $D_F$ and $b_F \in \mathbb{Z}^k$ such that

- $D_F \in \{0, 1, 2\}^{k \times m}$
- The sum of entries in any column of $D_F$ is exactly 2
- A fractional matroid matching $x$ is in $F$ iff $y_e = 0$ for $e \in S$ and $D_F x = b_F$.

## 6.2 A Quasi-NC Algorithm with Isolating Weight Assignment

In this section we will describe how we can construct an isolating weight assignment for fractional matroid matching with just the number of lines as input.

Now for a face $F$ of a polytope, let $\mathcal{L}_F$ denote the lattice

$$\mathcal{L}_F = \{v \in \mathbb{Z}^m \mid v = \alpha(x_1 - x_2) \text{ for some } x_1, x_2 \in F \text{ and } \alpha \in \mathbb{R}\}$$

and $\lambda(\mathcal{L}_F)$ denote the length of the shortest vector of $\mathcal{L}_F$. Hence $\mathcal{L}_F$ consists of all integral vectors parallel to the face $F$.

Now by Theorem 6.1.3 the face maximizing the size is described by the equation $D_F x = b_F$ where $D_F \in \{0,1,2\}^{k\times m}$ with column sum 2. Hence $\mathcal{L}_F$ is exactly the set of integral vectors in the null space of $D_F$. Therefore

$$\mathcal{L}_F = \{v \in \mathbb{Z}^m \mid D_F v = 0\}$$

So we will prove that the number of vectors in $\mathcal{L}_F$ with size less than twice the length of shortest vector is polynomially bounded in Subsection 1.2.2.

First we will show how we can interpret $D_F$ an incidence matrix for a graph instead of general matrix. $D_F \in \{0,1,2\}^{k\times m}$ where every column sum is 2. Hence we can thing of a graph $G_D$ with vertex set $[k]$ and the $m$ edges defined as follows: for every $e \in [m]$ the $e$-th edge of $G_D$ is drawn between the vertices $s,t \in [k]$ if $D_F[s,e] = D_F[t,e] = 1$ and $e$-th edge is a self look on the vertex $s \in [k]$ if $D_F[s,e] = 2$.

## 6.2.1 Alternating Circuits

First we will define some elements called alternating indicator vetors and alternating circuits following the [ST17] for the proof of Theorem 6.2.3.

**Definition 6.2.1: Alternating Indicator Vector & Alternating Circuit**

Let $C = v_0 \xrightarrow{e_0} v_1 \xrightarrow{e_1} v_2 \cdots \xrightarrow{e_{k-2}} v_{k-1} \xrightarrow{e_{k-1}} v_0$ be a closed walk of even length in a multigraph $G$ with loops. Then the *Alternating Indicator Vector* of $C$ denoted by $(\pm\mathbb{1})_C$ is the vector

$$(\pm\mathbb{1})_C \coloneqq \sum_{i=0}^{k-1}(-1)^i \mathbb{1}_{e_i}$$

$C$ is called *Altenating Circuit* if its alternating indicator vector is nonzero

We can use the parity for $(-1)^i$ as direction of movement in $C$. So a closed walk $C$ is a alternating circuti if there exists at least one edge $e \in C$ for which $C$ has moved through $e$ more time in one direction than the other.

**Observation.** $|(\pm\mathbb{1})_C| \le |C|$ *for any even length closed walk* $C$.

Now we will prove a property for all alternating circuits in $G_D$

**Lemma 6.2.1** [GOR24, Proof of Claim 1, Proof of Theorem 3.4]
For any alternating circuit $C$ we have $D_F \cdot (\pm\mathbb{1})_C = 0$

***Proof:*** Suppose $C = v_0 \xrightarrow{e_0} v_1 \xrightarrow{e_1} v_2 \cdots \xrightarrow{e_{k-2}} v_{k-1} \xrightarrow{e_{k-1}} v_0$. We denote the $i$-th column of $D_F$ is denoted by $D_i$. Now

$$D_i \cdot (\pm\mathbb{1})_C = \sum_{j=0}^{k-1}(-1)^j D_F[i,e_j] = \sum_{e_j : i \in e_j \in C}(-1)^j D_F[i,e_j]$$

Hence only the edges in $C$ which are incident on $i$ contributes to the above sum. Therefore the whole sum is partitioned into distinct subparts of walks where each subpart is of the form

$$v_s \xrightarrow{e_s} \underbrace{i \xrightarrow{e_{s+1}} i \xrightarrow{e_{s+2}} \cdots \xrightarrow{e_{s+k-1}} i}_{k \text{ times}} \xrightarrow{e_{s+k}} v_t \qquad \text{where } v_s, v_t \neq i$$

i.e. the part starts from a vertex not equal to $i$ then goes to $i$ and after looping in $i$ for some times the walk goes to another vertex not equal to $i$. We will show that for each of these parts the contribution to the sum is 0.

Now for such a subpart their contribution to the sum is

$$\delta = (-1)^s + 2\sum_{j=1}^{k-1}(-1)^{s+i} + (-1)^{s+k}$$

We will analyze case wise:

**Case 1: $k$ is odd:** Then $k-1$ is even. Hence

$$\sum_{j=1}^{k-1}(-1)^{s+i} = (-1)^s \sum_{j=1}^{k-1}(-1)^i = 0$$

Therefore

$$\delta = (-1)^s + (-1)^{s+k} = (-1)^s[1+(-1)^k] = 0$$

as $k$ is odd.
**Case 2: $k$ is even:** Then $k-2$ is even. Hence

$$\sum_{j=1}^{k-1}(-1)^{s+i} = \left[\sum_{j=1}^{k-2}(-1)^{s+i}\right] + (-1)^{s+k-1} = (-1)^{s+k-1}$$

Hence

$$\delta = (-1)^s + 2(-1)^{s+k-1} + (-1)^{s+k} = (-1)^s[1+2(-1)^{k-1}+(-1)^k] = 0$$

Therefore we showed that for each such parts their contribution to the sum is 0. Therefore the total sum is 0 i.e. $D_i \cdot (\pm\mathbb{1})_C = 0$. Since this is true for all $i \in [k]$ we have $D_F \cdot (\pm\mathbb{1})_C = 0$. ■

Now we will show that any vector in the lattice $\mathcal{L}_F$ can be decomposed into finite sum of alternating indicator vectors of alternating circuits with the property that for each such alternating circuit $C$, $|(\pm\mathbb{1})_C| = |C|$. Before that we introduce a relation between two vectors in $\mathbb{R}^m$ this will come in handy for the decomposition.

**Definition 6.2.2: Conformal**

For $x, y \in \mathbb{R}^m$, $x$ is said to be conformal to $y$ if $x_i y_i \geq 0$ and $|x_i| \leq |y_i|$ $\forall\, i \in [m]$ and it is denoted by $x \sqsubseteq y$

**Lemma 6.2.2** [GOR24, Claim 1, Proof of Theorem 3.4]
For any $x \in \mathcal{L}_F$, $\exists$ alternating circuits $C_1, C_2, \ldots, C_t$ in $G_D$ such that

$$x = \sum_{i=1}^{t}(\pm\mathbb{1})_{C_i}$$

where $\forall\, i \in [t]$, $(\pm\mathbb{1})_{C_i} \sqsubseteq x$ and $|(\pm\mathbb{1})_{C_i}| = |C_i|$.

***Proof:*** We will decompose a given $x$ into alternating indicator vectors by the following iterative algorithm:

**Algorithm 1:** Decomposition of a Lattice Vector

```
Input: $x \in \mathcal{L}_F$
Output: $\mathcal{Y} = \{y_i\}$ where $|\mathcal{Y}| < \infty$ and $\forall\, y_i \in \mathcal{Y}$ are alternating indicator vectors
1  begin
2      while $|x| \neq 0$ do
3          $y \longleftarrow 0$, $j \longleftarrow 1$
4          Let $e_0 \in [m]$ such that $x_{e_0} > 0$
5          $y_{e_0} \longleftarrow 1$ and let $e_0$-th edge in $G_D$ be $\{v_0, v_1\}$
6          while True do
7              if $\exists\, e \in [m]$ such that $e$-th edge is $\{v_j, u\}$, $|x_e| > |y_e|$ and $(-1)^j x_e > 0$ then
8                  $y_e \longleftarrow y_e + (-1)^j$, $e_j \longleftarrow e$
9                  $v_{j+1} \longleftarrow u$
10                 $j \longleftarrow j+1$
11             else
12                 return $x \notin \mathcal{L}_F$
13             if $j \equiv 0 \pmod 2$ and $v_j = v_0$ then
14                 $x \longleftarrow x - y$
15                 return $y$ and exit inner while loop
```

Now suppose $x$ denote the vector at some iteration of the outer while loop. Now forall $e \in [m]$, let at $j$th and $l$th iteration of the inner while loop $(-1)^j$ and $(-1)^l$ are added to $y_e$ respectively. Now both $j$ and $l$ has same parity because since the edge is not changing both times $(-1)^j x_e > 0$ and $(-1)^l x_e > 0$ has to be satisfies. Therefore we get $j$ and $l$ have same parity. Hence for a full run of the inner while loop for any $e \in [m]$ everytime $y_e$ is changed the same $(-1)^j$ is added. Hence whenever $y_e$ is changed $|y_e|$ is increased. Therefore $|y|$ increases for each iteration of the inner while loop. But $|y|$ cannot exceed $|x|$ since as the addition step works when $\exists\ e \in [m]$ with $|x_e| > |y_e|$ and $(-1)^j x_e > 0$, after addition $|y_e + (-1)^j| \le |y_e| + 1 \le |x|$. So if we start with $|y| \le |x|$ after addition with each iteration of inner while loop we still have $|y| \le |x|$. Also since for every such edge we add $(-1)^j$ to $y_e$ when $(-1)^j x_e > 0$ and $|x_e| > |y_e|$. So by the first condition we have $y_e$ and $x_e$ has the same sign i.e. $x_e y_e > 0$. And we also have $|x_e| \ge |y_e|$. Therefore we have $y \sqsubseteq x$.

Now since we start the algorithm by adding 1 to $y_{e_0}$ at Line 5 we have initially $|y| > 0$ and afterwards with each iteration of the inner while loop $|y|$ increases so we have $|y| > 0$.Now if the current iteration of the inner while loop ends at Line 13 then we get a closed walk $C$ since in each iteration of the inner while loop the algorithm follows a walk in $G_D$. So $C$ is an alternating circuit and we have $|y| = |C|$. Now since $y \sqsubseteq x$ we have $|x - y| = |x| - |y| < |x|$. Since $C$ is alternating circuit by Lemma 6.2.1 we have $y \in \mathcal{L}_F$. Hence we have $x - y \in \mathcal{L}_F$.

Now all that remains is to show that the algorithm never goes to Line 11 if $x \in \mathcal{L}_F$. The only reason the algorithm can go to Line 11 if at some iteration of the inner while loop can not find an edge $e \in [m]$ such that $|x_e| > |y_e|$ and $(-1)^j x_e > 0$ where $e \in \delta(v_j)$ where $\delta(v_j)$ denotes te set of edges incident on $v_j$. Suppose at this point we have the walk

$$\mathcal{P} = v_0 \xrightarrow{e_0} v_1 \xrightarrow{e_1} \cdots \xrightarrow{e_{j-1}} v_j$$

Now by following the proof of Lemma 6.2.1 we have that $D_i y = 0$ forall $i \in [k]$ except $i \notin \{v_0.v_j\}$.

***Claim:*** If $j$ is odd then $D_{v_j} y > 0$ and if $j$ is even then $D_{v_j} y < 0$.
***Proof:*** We will prove the case for $j$ is odd. The even case will follow similarly. Since $j$ is odd we have $j - 1$ even. Therefore $x_{e_{j-1}} > 0$ as $(-1)^{j-1} x_{e_{j-1}} > 0$. Therefore for both $e_0$ and $e_{j-1}$, $x_{e_0}, x_{e_{j-1}}$ positive. Now take the partitions as described in the proof of Lemma 6.2.1. Now we analyze case wise:
**Case 1:** $v_0 \ne v_j$**:** Then only the partition containing $v_{e_j}$ might contribute something nonzero because other partitions contributes 0 to the sum $D_{v_j} y$. Let the partition be

$$v_s \xrightarrow{e_{j-k-1}} \underbrace{v_j \xrightarrow{e_{j-k}} v_j \xrightarrow{e_{j-k+1}} \cdots \xrightarrow{e_{j-1}} v_j}_{k+1 \text{ times}} \qquad \text{where } v_s \ne v_j$$

Now $k$ can not be more than 1 since if $e_{j-1}$ and $e_{j-2}$ are both loops in $v_j$ then for $e_{j-1}$, $x_{e_{j-1}} > 0$ but $x_{e_{j-2}} < 0$ but $e_{j-1} = e_{j-2} \implies 0 < x_{e_{j-1}} = x_{e_{j-2}} < 0$. Then this partition contributes $\delta = (-1)^{j-2} + 2(-1)^{j-1} = -1 + 2 > 0$ to $D_{v_j} y$ if there is a loop and if there is no loop then it also contributes 1 as then the contribution will be only $(-1)^{j-1} = 1 > 0$.
**Case 2:** $v_0 = v_j$**:** If $v_0$ and $v_j$ is in same partition i.e. $j = 1$ then the it contributes $(-1)^0 = 1 > 0$ to the sum. Otherwise $v_0$ is in different partition. By the above case we know that the value contributed by the partition containing $v_j$ is 1. So we have to only find the contribution by the partition containing $v_0$. Again by same logic as the above case from $v_0$ at most one loop starting from $v_0$ and then it moves to some other vertex. Hence the sum contributed is $2(-1)^0 + (-1)^1 = 2 - 1 = 1$ if there is a loop and if there is no loop then it contributes $(-1)^0 = 1$. Hence in both cases the contribution of the partition containing $v_0$ to the sum $D_{v_j} y$ is 1. Hence we have $D_{v_j} y > 0$.

Hence by above we get that if $j$ is odd then the value $D_{v_j} y > 0$. Similarly following the same process in the case of $j$ is even we get $D_{v_j} y < 0$. ■

Hence by the lemma we have that if $j$ is odd $D_{v_j} y > 0$ and if $j$ is even then $D_{v_j} y < 0$. So WLOG suppose $j$ is odd. Now define

$$\text{supp}^+(x) = \{i \in [m] \mid x_i > 0\} \qquad \text{and} \qquad \text{supp}^-(x) = \{i \in [m] \mid x_i < 0\}$$

Then we have

$$0 = D_{v_j} x = \sum_{e \in \text{supp}^+(x) \cap \delta(v_j)} D_F[v_j, e] \cdot |x_e| - \sum_{e \in \text{supp}^-(x) \cap \delta(v_j)} D_F[v_j, e] \cdot |x_e|$$
$$0 < D_{v_j} y = \sum_{e \in \text{supp}^+(x) \cap \delta(v_j)} D_F[v_j, e] \cdot |y_e| - \sum_{e \in \text{supp}^-(x) \cap \delta(v_j)} D_F[v_j, e] \cdot |y_e|$$

Now be construction of $y$ we have $y \sqsubseteq x \implies |x_e| \geq |y_e|, x_e y_e > 0 \forall\, e \in [m]$. The algorithm stopped at Line 13 since $\nexists\, e \in \delta(v_j)$ such that $|x_e| > |y_e|$ and $(-1)^j x_e = -x_e > 0$. Therefore $|x_e| = |y_e|$ for all $e \in \mathrm{supp}^+(x) \cap \delta(v_j)$ otherwise the algorithm would have proceeded one more step and $\mathrm{supp}^-(x) \cap \delta(v_j) = \mathrm{supp}^-(y) \cap \delta(v_j)$ since for any $e \in [m], x_e y_e > 0$. Therefore $\mathrm{supp}^+(x) \cap \delta(v_j) = \mathrm{supp}^+(y) \cap \delta(v_j)$. Now for all $e \in \mathrm{supp}^+(x) \cap \delta(v_j)$ we have $|x_e| \geq |y_e|$. Hence we have

$$0 = D_{v_j} y \geq D_{v_j} y > 0$$

It is a contradiction. Hence if $x \in \mathcal{L}_F$ the algorithm never goes to Line 13. And therefore the algorithm successfully decomposes $x$ into sum of alternating circuits. ■

### 6.2.2 Bounding vectors in $\mathcal{L}_F$ with Small Size

**Theorem 6.2.3** [GOR24]
Let $D \in \{0,1,2\}^{pimesm}$ be a matrix such that the sum of entries of each column equals 2. Let $\mathcal{L}_D$ denote the lattice $\{v \in \mathbb{Z}^m \mid Dv = 0\}$. Then it holds that

$$|\{v \in \mathcal{L}_D \mid |v| < 2\lambda(\mathcal{L}_D)\}| \leq m^{O(1)}$$

***Proof:*** For the given $D$ consider the graph $G_D$ obtained from $D$ as explained at the start of Section 1.2. to show that the number of vectors in $\mathcal{L}_D$ with size less than twice the size of shortest vector in $\mathcal{L}_D$ we will show that for any such lattice vector there is only one alternating circuit in the decomposition of the vector in Lemma 6.2.2.

***Claim:*** Any lattice vector $x \in \mathcal{L}_D$ with $|x| < 2\lambda(\mathcal{L}_D)$ is an alternating vector $(\pm\mathbb{1})_C$ of some alternating circuit $C$ in $G_D$ such that $|x| = |C|$

***Proof:*** Suppose the contrary. Since $x \in \mathcal{L}_D$ by Lemma 6.2.2 $\exists\, C_1, \ldots, C_t$ with $t \geq 2$ such that $x = \sum_{i=1}^{t} (\pm\mathbb{1})_C$ with $(\pm\mathbb{1})_C \sqsubseteq x$ and $|(\pm\mathbb{1})_{C_i}| = |C_i|$ for all $i \in [t]$. Then we have

$$|x| = \sum_{i=1}^{t} |(\pm\mathbb{1})_{C_i}| \geq t\lambda(\mathcal{L}_D) \geq 2\lambda(\mathcal{L}_D)$$

which is a contradiction since we assumed that $|x| < 2\lambda(\mathcal{L}_D)$. Hence $t = 1$ i.e. $x = (\pm\mathbb{1})_C$ for some alternating circuit $C$ with $|x| = |C|$. ■

Hence the Claim implies that $\lambda(\mathcal{L}_D)$ is equal to the size of the smallest alternating circuit of $G_D$. And it also implies that we only need to bound the number of alternating indicator vectors that correspond to alternating circuit of size at most $2\lambda(\mathcal{L}_D)$ to prove the lemma. For that by the Theorem 6.2.4 we get that the number of such alternating indicator vectors are polynomially bounded by $m$. ■

In the following theorem I have modified the proof of the theorem since we are not working the general setting like in [ST17]. We are basically taking node-weight for every vertex to be 1.

**Theorem 6.2.4** [ST17, Lemma 5.4]
Let $G$ be a graph on $n$ vertices such that the size of the smallest alternating circuit $\lambda$. Then the cardinality of the set

$$\{(\pm\mathbb{1})_C \mid C \text{ is an alternating circuit in } G \text{ of size at most } 2\lambda\}$$

is at most $n^{17}$.

***Proof:*** Like in the proof of Lemma 2.2.4 we will associate a small signature $\sigma(C)$ with each alternating circuit $C$ in $G$ of size at most $2\lambda$. The signatures have the property that alternating circuits with different alternating indicator vectors are assigned to different signatures. This will prove that the number of alternating circuits in $G$ of size at most $2\lambda$ is at most the number of possible signatures which we will show polynomially bounded.

So let

$$C = v_0 \xrightarrow{e_0} v_1 \xrightarrow{e-1} \cdots \xrightarrow{e_{k-2}} v_{k-1} \xrightarrow{e_{k-1}} v_0$$

be an alternating circuit in $G$ of size at most $2\lambda$. Now for $v_i \in C$ define

$$in(v_i) \coloneqq \text{Incoming edge of } v_i \qquad out(v_i) \coloneqq \text{Ourgoing edge og } v_i$$

Now we define the signature $\sigma(C)$:

- Let $i_0 = 0$ be the first vertex in $C$.
- For $j \in [3]$, select $i_j = \left\lceil \frac{j\lambda}{4} \right\rceil$. Hence $\sum\limits_{l=i_{j-1}+1}^{i_j-1} 1 \leq \frac{\lambda}{2}$.
- Take $t = \max\{j \mid i_j < k\}$. Output

$$\sigma(C) = \Big( (-1)^{i_j}, in(v_{i_j}), out(v_{i_j}) \Big)_{j \in \{0,1,\ldots,t\}}$$

**Note:-**

We are considering $t$ as an important index because size of $C$ can be small so that $k < i_3 < 2k$ then to define $i_3$ properly we basically take $v_{i_j} = v_{i_j \bmod k}$. In that case the index which is still less than $k$ helps us to consider the cycle as joining of $t$ different paths. Also this case can only happen to $i_3$ since $C$ has length at least $\lambda$ so both $i_1$ and $i_2$ are same after modulo $k$.

The indices $i_0, i_1, \ldots, i_t$ partition $C$ into paths:

$$\begin{aligned}
C_0 &= v_{i_0} \xrightarrow{e_{i_0}} v_{i_0+1} \xrightarrow{e_{i_0+1}} \cdots \xrightarrow{e_{i_1-1}} v_{i_1} \\
C_1 &= v_{i_1} \xrightarrow{e_{i_1}} v_{i_1+1} \xrightarrow{e_{i_1+1}} \cdots \xrightarrow{e_{i_2-1}} v_{i_2} \\
C_2 &= v_{i_2} \xrightarrow{e_{i_2}} v_{i_2+1} \xrightarrow{e_{i_2+1}} \cdots \xrightarrow{e_{i_3-1}} v_{i_3} \\
&\vdots \\
C_t &= v_{i_t} \xrightarrow{e_{i_t}} v_{i_t+1} \xrightarrow{e_{i_t+1}} \cdots \xrightarrow{e_{i_0-1}} v_{i_0}
\end{aligned}$$

So the node-weight of internal vertices for each path is $C_i$ for $i \in \{0, 1, \ldots, t-1\}$ is at most $\frac{\lambda}{2}$.Therefore by the maximality of $t$ we have:

$$\underbrace{\sum_{l=i_0+1}^{i_1} 1}_{\geq \frac{\lambda}{2}} + \underbrace{\sum_{l=i_1+1}^{i_2} 1}_{\geq \frac{\lambda}{2}} + \cdots + \underbrace{\sum_{l=i_{t-1+1}^{i_t} 1} 1}_{\geq \frac{\lambda}{2}}$$

Each sum is at least $\frac{lm}{2}$ so we have the total node-weight of internal vertices of $C_t$ is at most $\frac{\lambda}{2}$.

Now we will count the number of possible signatures. For each $j \in \{0, 1, \ldots, t\}$ in the tuple $\Big( (-1)^{i_j}, in(v_{i_j}), out(v_{i_j}) \Big)$ there are 2 possible parity of $i_j$ and both $in(v_{i_j})$ and $out(v_{i_j})$ has at most $n^2$ choices. So for each $j$ there are $2n^4$ many possible tuples. Therefore:

For $t = 0$ there are $2n^4$ many signatures
For $t = 1$ there are $(2n^4)^2$ many signatures
For $t = 2$ there are $(2n^4)^3$ many signatures
For $t = 3$ there are $(2n^4)^4$ many signatures

Hence total number of possible signatures is $2n^4 + (2n^4)^2 + (2n^4)^3 + (2n^4)^4 < n^{17}$. So number of signatures is polynomially bounded. Now we will show that any 2 alternating circuits $C, D$ of size at most $2\lambda$ have different signatures i.e.

$$\sigma(C) \neq \sigma(D) \iff (\pm\mathbb{1})_C \neq (\pm\mathbb{1})_D$$

***Claim:*** $C$ is the only alternating circuit $C$ of size at most $2\lambda$ that is associated to the signature $\sigma(C)$.
***Proof:*** So let $C, D$ are two alternating circuits of size at most $2\lambda$ and $\sigma(C) = \sigma(D)$. As described above $C$ is partitioned into paths $C_0, \ldots, C_t$ using $i_0, \ldots, i_t$ and similarly $D$ is also partitioned into $D_0, \ldots, D_t$ using $j_0, \ldots, j_t$. Since $\sigma(C) = \sigma(D)$, $i_k$ and $j_k$ have same parity for all $k \in \{0, 1, \ldots, t\}$. and since $in(v_{i_k}) = in(v_{j_k})$ and $out(v_{i_k}) = out(v_{j_k})$ we have $v_{i_k} = v_{j_k}$ for all $k \in \{0, 1, \ldots, t\}$. Now let $b_k$ denote the parity of $k$th tuple in $\sigma(C)$ i.e. $b_k = (-1)^{i_k} = (-1)^{j_k}$. Then we have

$$(\pm\mathbb{1})_C = \sum_{k=0}^{t} b_k (\pm\mathbb{1})_{C_k} \qquad (\pm\mathbb{1})_D = \sum_{k=0}^{t} b_k (\pm\mathbb{1})_{D_k}$$

Since we know $(\pm\mathbb{1})_C \neq (\pm\mathbb{1})_D$, $\exists\ k \in \{0, 1, \ldots, t\}$ such that $(\pm\mathbb{1})_{C_k} \neq (\pm\mathbb{1})_{D_k}$Now basically we will glue $C_k$ and $D_k$ togather. Now

$$\begin{tikzcd}[row sep=small] C_k = v_{i_k} \arrow[d, phantom, "\parallel"] = a \arrow[r] & b \arrow[r] & \cdots \arrow[r] & c \arrow[r] & d = v_{i_{k+1 \bmod t}} \arrow[d, phantom, "\parallel"] \\ v_{j_k} & & & & v_{j_{k+1 \bmod t}} \end{tikzcd}$$

Hence the path $C_k$ and $D_k$ differs in between the path $b \rightsquigarrow c$. Let $P_C$ denotes the path from $b$ to $c$ following $C_k$ and $P_D$ denotes the path from $b$ to $c$ following $D_k$. As $(-1)^{i_k} = (-1)^{j_k}$ and $(-1)^{i_{k+1 \bmod t}} = (-1)^{j_{k+1 \bmod t}}$. Therefore we have

$$|C_k| + |D_k| \equiv 0 \bmod 2 \implies |P_C| + |P_D| \equiv 0 \bmod 2$$

So denote the cycle $C$ to be the closed walk:

$$b \xrightarrow{f_1} \cdots \xrightarrow{f_{|P_C|}} c \xrightarrow{g_1} \cdots \xrightarrow{g_{|P_D|}} b$$

Where $f_1, \ldots, f_{|P_C|}$ are the edges of the path $P_C$ and $g_1, \ldots, g_{|P_D|}$ are the edges of the path $P_D$ in reverse. Hence length of the cycle $C$ is

$$|P_C| + |P_D| = |C_k| - 2 + |D_k| - 2 < \frac{\lambda}{2} + \frac{\lambda}{2} = \lambda$$

Therefore if $B$ is an alternating circuit then we have an alternating circuit $(\pm\mathbb{1})_C$ of size less than $\lambda$, which is not possible. Hence all that is left is showing that $(\pm\mathbb{1})_C$ is indeed an alternating circuit i.e. $(\pm\mathbb{1})_C \neq 0$. Now:

$$\begin{aligned} -(\pm\mathbb{1})_C &= \sum_{i=1}^{|P_C|} (-1)^i \mathbb{1}_{f_i} + \sum_{i=1}^{|P_D|} (-1)^{|P_C|+i} \mathbb{1}_{g_i} \\ &= \underbrace{\left[ (-1)^0 \mathbb{1}_{out(a)} + \sum_{i=1}^{|P_C|} (-1)^i \mathbb{1}_{f_i} + (-1)^{|P_C|+1} \mathbb{1}_{in(d)} \right]}_{=(\pm\mathbb{1})_{C_k}} \\ &\qquad - \underbrace{\left[ (-1)^{|P_C|+1} \mathbb{1}_{in(d)} + \sum_{i=1}^{|P_D|} (-1)^{|P_C|+i+2} \mathbb{1}_{g_i} + (-1)^{|P_C|+|P_D|+3} \mathbb{1}_{out(a)} \right]}_{=-(\pm\mathbb{1})_{D_k}} \end{aligned}$$

Therefore we have $(\pm\mathbb{1})_C = (\pm\mathbb{1})_{D_k} - (\pm\mathbb{1})_{C_k} \neq 0$. Therefore we have $(\pm\mathbb{1})_C$ is an alternating circuit. This leads to a contradiction ■

So there aren't two different alternating circuits of size at most $2\lambda$ with same signatures. Therefore number of alternating circuits in $G$ of size at most $2\lambda$ is bounded by number of signatures which is at most $n^{17}$. ■

### 6.2.3 Algorithm for Finding Isolating Weight Assignment

With this theorem we have

**Theorem 6.2.5** [GTV21, Theorem 2.5]
Let $k$ be a positive integer and $P \subseteq \mathbb{R}^m$ a polytope such that its extreme ppoints are in $\left\{0, \frac{1}{k}, \frac{2}{k}, \ldots, 1\right\}^m$ and there exists a constant $c > 1$ with

$$|\{v \in \mathcal{L}_F \colon |v| < c\lambda(\mathcal{L}_F)\}| \leq m^{O(1)}$$

for any face $F$ of $P$. Then there exists an algorithm that, given $k$ and $m$, outputs a set $\mathcal{W} \subseteq \mathbb{Z}^m$ of $m^{O(\log km)}$ weight assignments with weights bounded by $m^{O(\log km)}$ such that there exists at least one $w \in \mathcal{W}$ that is isolating for $P$, in time $polylog(km)$ using $m^{O(\log km)}$ many parallel processors.

Using this we finally have an algorithm for isolating a fractional matroid matching polytope:

**Theorem 6.2.6** [GOR24, Theorem 3.1]
There exists an algorithm that given $m \in \mathbb{Z}_+$ outputs a set $\mathcal{W} \subseteq \mathbb{Z}_+^m$ of $m^{O(\log m)}$ weight assignments with weights bounded by $m^{O(\log m)}$ such that, for any fractional matroid matching polytope $P$ of $m$ lines, there exists at least one $w \in \mathcal{W}$ that is isolating for $P$, in time $polylog(m)$ usign $m^{O(\log m)}$ many parallel processors.

Chapter 7

# Isolation of Paths in Layered Graph

[vMP19] In this chapter we will restrict our attention to layered digraphs. Here for any $k \in \mathbb{Z}_{\geq 0}$ we denote $[\![k]\!] = \{0, 1, \ldots, k\}$.

**Definition 7.0.1: Layered Digraph**

For $n = |V|$ we have $V \subseteq [n] \times [\![d]\!]$ and $E \subseteq \bigcup_{i \in [d]} (V_{i-1} \times V_i)$ for $i \in [d]$ with $V_i := V \cap [n] \times \{i\}$ i.e. the $i$th layer in $G$

Basically a layered graph $G = (V, E)$ of depth $d$ consists of $d + 1$ layers of vertices such that the edges only go from one layer to the next.

Chapter 8

# Bibliography

[BCP83] A. Borodin, S. Cook, and N. Pippenger. Parallel computation for well-endowed rings and space-bounded probabilistic machines. *Information and Control*, 58(1–3):113–136, July 1983.

[Ber84] Stuart J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Information Processing Letters*, 18(3):147–150, March 1984.

[CRS93] Suresh Chari, Pankaj Rohatgi, and Aravind Srinivasan. Randomness-Optimal Unique Element Isolation, with Applications to Perfect Matching and Related Problems. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, STOC '93, page 458–467. ACM Press, June 1993.

[FGT16] Stephen Fenner, Rohit Gurjar, and Thomas Thierauf. Bipartite Perfect Matching is in quasi-NC. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, STOC '16. ACM, June 2016.

[GG17] Shafi Goldwasser and Ofer Grossman. Bipartite Perfect Matching in Pseudo-Deterministic NC. 2017.

[GOR24] Rohit Gurjar, Taihei Oki, and Roshan Raj. Fractional Linear Matroid Matching is in quasi-NC. 2024.

[GP13] Dion Gijswijt and Gyula Pap. An algorithm for weighted fractional matroid matching. *Journal of Combinatorial Theory, Series B*, 103(4):509–520, July 2013.

[GTV21] Rohit Gurjar, Thomas Thierauf, and Nisheeth K. Vishnoi. Isolating a Vertex via Lattices: Polytopes with Totally Unimodular Faces. *SIAM Journal on Computing*, 50(2):636–661, January 2021.

[KS01] Adam R. Klivans and Daniel Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, STOC01. ACM, July 2001.

[LP86] László Lovász and M.D. Plummer. *Matching Theory*. Number 121 in North-Holland Mathematics Studies. North-Holland, Amsterdam, 1986.

[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 *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, STOC '00. ACM, May 2000.

[MVV87] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as Easy as Matrix Inversion. In *Proceedings of the nineteenth annual ACM conference on Theory of computing - STOC '87*, STOC '87. ACM Press, 1987.

[Sch80] J. T. Schwartz. Fast Probabilistic Algorithms for Verification of Polynomial Identities. *Journal of the ACM*, 27(4):701–717, October 1980.

[ST17] Ola Svensson and Jakub Tarnawski. The Matching Problem in General Graphs Is in quasi-NC. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, October 2017.

[TK92] C. P. Teo and K. M. Koh. The number of shortest cycles and the chromatic uniqueness of a graph. *Journal of Graph Theory*, 16(1):7–15, mar 1992.

[vMP19] Dieter van Melkebeek and Gautam Prakriya. Derandomizing Isolation in Space-Bounded Settings. *SIAM Journal on Computing*, 48(3):979–1021, January 2019.

[VV92] John H Vande Vate. Fractional Matroid Matchings. *Journal of Combinatorial Theory, Series B*, 55(1):133–145, May 1992.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Lecture Notes in Computer Science*, pages 216–226. Springer Berlin Heidelberg, 1979.