1 Outline and Background

In this lecture, we introduce algebraic methods to find a perfect matching in a graph. In particular, we use so-called the "polynomial method" which is based on the fact that low degree polynomials have a few number of roots. An interesting aspect of this technique is that not only it finds a perfect matching, but it is also useful in finding other kinds of structures. For example, it can be used to find a *Red-Blue perfect matching*. That is, when we are given a graph G whose edges are colored in either red or blue and a fixed number k, we would like to find a perfect matching M containing exactly k red edges. It is still an open question whether there is a deterministic polynomial time algorithm to solve this problem. We introduce a randomized algorithm for Red-Blue perfect matching using the polynomial method later in this note.

Below are some know results for perfect matching problem. These algorithms all work for general graphs.

Authors	**Complexity**	**Comments**
Lovász [Lov79]	$O(m \cdot n^{\omega})$	Randomized, via Tutte and Edmonds matrix
Micali and Vazirani [MV80]	$O(m \cdot n^{1/2})$	
Mulmuley, Vazirani, and Vazirani [MVV87]	$O(m \cdot n^{\omega})$	Parallel algorithm
Rabin and Vazirani [RV89]	$O(n \cdot n^{\omega})$	
Mucha and Sankowski [MS04]	$O(n^{\omega})$	

We look at matrix-based algorithms for finding a perfect matching (PM) or reporting its non-existence. In Section 2, we cover basic facts about low degree polynomials. In Section 3, we introduce an $O(m \cdot n^{\omega})$ time algorithm using the Edmonds matrix for bipartite matching problem and the Tutte matrix for non-bipartite matching problem due to Lovász [Lov79]. In Section 4, we show an $O(n \cdot n^{\omega})$ time algorithm given by Rabin and Vazirani [RV89]. Lastly, in Section 5, we cover an application of these methods to Red-Blue matching problem, and we explain the so-called "isolation lemma" introduced by Mulmuley, Vazirani, and Vazirani [MVV87].

2 Preliminaries: roots of low degree polynomials

An important idea for today's lecture is that low degree polynomials have a few roots. In this section, we will see this for both univariate and multivariate polynomials. The following theorem is a consequence of the fundamental theorem of algebra, and it is for univariate polynomials.

Theorem 9.1 (Fundamental theorem of algebra)**.** *An univariate polynomial $p(x)$ of degree at most d over any field has at most d roots (unless $p(x)$ is zero polynomial).*

Now, let us look at multivariate polynomials. Can we say that a low degree polynomial which is multivariate has a small number of roots as we did in Theorem 9.1 for univariate polynomials? For example, $p(x, y) = xy$ has degree 2 and the solutions to $p(x, y) = 0$ are exactly the points in $\{(x, y) \in \mathbb{R}^2 : \ x = 0 \text{ or } y = 0\}$. Observe that $\{(x, y) \in \mathbb{R}^2 : \ x = 0 \text{ or } y = 0\}$ is very sparse in $\mathbb{R}^2$. Let us formalize this observation. The following result is called Schwartz-Zippel lemma.

Theorem 9.2 (DeMillo and Lipton [DL78], Zippel [Zip79], Schwartz [Sch80])**.** *Let $p(x_1, \ldots, x_n)$ be a non-zero polynomial of degree at most d over a field $\mathbb{F}$. Suppose we choose values $R_1, \ldots, R_n$ independently and uniformly at random from $S \subseteq \mathbb{F}$. Then*

$$\mathbb{P}\left[p(R_1, \ldots, R_n) = 0\right] \leq \frac{d}{|S|}.$$

and this implies the number of roots of p in S^n is at most $d|S|^{n-1}$.

Proof. We argue by induction on n.

Base case ($n = 1$): By Theorem 9.1, any univariate polynomial with degree d has at most d roots. Thus we have that $\mathbb{P}\left[p(R_1) = 0\right] \leq d/|S|$ as desired.

Inductive step: Let k be the highest power of x_n that appears in p and let $q(x_1, \ldots, x_{n-1})$ and $r(x_1, \ldots, x_n)$ be the (unique) polynomials such that $p(x_1, \ldots, x_n) = x_n^k q(x_1, \ldots, x_{n-1}) + r(x_1, \ldots, x_n)$. We also assume that the highest power of x_n that appears in r is less than k. That is, when dividing p by x_n^k, q is the quotient and r is the remainder. Now letting E be the event that $q(R_1, \ldots, R_{n-1})$ is zero we find

$$\begin{aligned}\mathbb{P}\left[p(R_1, \ldots, R_n) = 0\right] &= \mathbb{P}\left[p(R_1, \ldots, R_n) = 0 \mid E\right]\mathbb{P}\left[E\right] + \mathbb{P}\left[p(R_1, \ldots, R_n) = 0 \mid \bar{E}\right]\mathbb{P}\left[\bar{E}\right] \\ &\leq \mathbb{P}\left[E\right] + \mathbb{P}\left[p(R_1, \ldots, R_n) = 0 \mid \bar{E}\right]\end{aligned}$$

By the inductive assumption, and noting that q has degree at most $d - k$, we know $\mathbb{P}\left[E\right] = \mathbb{P}\left[q(R_1, \ldots, R_{n-1}) = 0\right] \leq (d-k)/|S|$ and similarly, viewing p as a polynomial only on x_n (with degree k), we know $\mathbb{P}\left[p(R_1, \ldots, R_n) = 0 \mid \bar{E}\right] \leq k/|S|$. Thus we get

$$\mathbb{P}\left[p(R_1, \ldots, R_n) = 0\right] \leq \frac{d-k}{|S|} + \frac{k}{|S|} = \frac{d}{|S|}.$$

□

Remark 9.3. Let $p(x_1, \ldots, x_n)$ be a multivariate polynomial of degree at most d over a field $\mathbb{F}$. Choose $R_1, \ldots, R_n$ independent and uniformly at random from $S \subseteq \mathbb{F}$ such that $|S| \geq dn^2$. If p is a non-zero polynomial,

$$\mathbb{P}\left[p(R_1, \ldots, R_n) = 0\right] \leq \frac{1}{n^2}.$$

If p is zero polynomial,

$$\mathbb{P}\left[p(R_1, \ldots, R_n) = 0\right] = 1.$$

3 Lovász's $O(m \cdot n^\omega)$ time algorithm

In this section, we explain an algorithm due to Lovász [Lov79] for finding a perfect matching in both bipartite and non-bipartite graphs.

3.1 Bipartite matching

Before we state the algorithm, we define the *Edmonds matrix*.

Definition 9.4. We are given a bipartite graph $G = (L \cup R, E)$ with $|L| = |R| = n$. The *Edmonds matrix* $\varepsilon(G)$ of G is defined as follows.

$$\varepsilon_{i,j} = \begin{cases} 0 & \text{if } (i,j) \notin E \text{ and } i \in L, j \in R \\ x_{i,j} & \text{if } (i,j) \in E \text{ and } i \in L, j \in R. \end{cases}$$

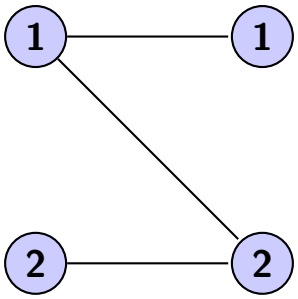

Figure 9.1: Bipartite graph

Example 9.5. Consider the bipartite graph in Figure 9.1. The Edmonds matrix of this graph is as follows.

$$\varepsilon = \begin{bmatrix} x_{11} & x_{12} \\ 0 & x_{22} \end{bmatrix}$$

Notice that the determinant of ε is exactly $x_{11}x_{22}$.

In general, we can use the Leibniz formula to compute the determinant of the Edmonds matrix of a bipartite graph G.

$$\det(\varepsilon(G)) = \sum_{\sigma \in S_n} (-1)^{\operatorname{sgn}(\sigma)} \prod_{i=1}^{n} \varepsilon_{i,\sigma(i)}$$

We remark that a perfect matching in G corresponds to a permutation $\sigma \in S_n$. Each $i \in L$ is matched to $\sigma(i) \in R$.

Proposition 9.6. *Let G be a bipartite graph, and let $\varepsilon(G)$ denote the Edmonds matrix of G. Then $\det(\varepsilon(G))$ is a non-zero polynomial if and only if G contains a perfect matching.*

Based on Proposition 9.6, we can provide a randomized algorithm to test whether a given bipartite graph contains a perfect matching.

Algorithm 1 PM-tester(G, S)

construct the Edmonds matrix $\varepsilon(G)$ of a bipartite graph G
for each non-zero entry ε_{ij} in $\varepsilon(G)$, we sample a number $R_{ij} \in S$ independently and uniformly at random
evaluate the determinant of $\varepsilon(G)$ with the chosen values for its entries
if $\det(\varepsilon(G)) = 0$ **then**
    return G does not have a perfect matching (NO)
else
    return G contains a perfect matching (YES)
end if

Lemma 9.7. *Assume that $|S| \geq n^3$. Then Algorithm 1 always returns 'NO' if G has no perfect matching, while it says 'YES' with probability at least $1 - \frac{1}{n^2}$ otherwise.*

Proof. It is straightforward by Theorem 9.2. □

Remark 9.8. We can compute the determinant of $\varepsilon(G)$ in $O(n^3)$ time using Gaussian elimination method, so Algorithm 1 runs in time $O(n^3)$. In fact, Bunch and Hopcroft [BH74] proved that both computing matrix inversion and determinant are as hard as matrix multiplication. Thus, we can make Algorithm 1 terminate in time $O(n^\omega)$.

Using the perfect matching tester described as Algorithm 1, we now have an algorithm to find a perfect matching in a graph if one exists. Suppose that a bipartite graph G has a perfect matching. Then we pick an edge e and check if $G[E-e]$, the induced subgraph of G by the edges in $E-e$, contains a perfect matching. If not, then e must be part of every perfect matching in G. The following is an algorithm based on this observation.

Algorithm 2 Find-PM(G, S)

```
if PM-tester(G, S) returns 'NO' then
    return G does not have a perfect matching
else
    let e = {u, v} be an edge in G
    if PM-tester(G[E − e], S) returns 'YES' then
        return Find-PM(G[E − e], S)
    else
        let Find-PM(G[V − {u, v}], S) return M′
        return M′ + e
    end if
end if
```

Theorem 9.9. *Given a bipartite graph G, Algorithm 2 finds a perfect matching with probability at least $\frac{1}{2}$ if G contains a perfect matching. In addition, Algorithm 2 runs in time $O(m \cdot n^\omega)$.*

Proof. At each recursive step, we either delete an edge or two vertices from the input graph. Thus, the number of total recursive steps inside Algorithm 2 is at most m. Hence, the running time of this algorithm is $O(m \cdot n^\omega)$. At each step, the probability that the PM-tester returns a wrong answer is at most $\frac{1}{n^2}$. By union bound, the probability that the PM-tester makes no mistake is at least $1 - \frac{m}{n^2}$ which is at least $\frac{1}{2}$. □

Corollary 9.10. *Given a bipartite graph G containing a perfect matching, there is an $\widetilde{O}(m \cdot n^\omega)$ time algorithm which finds a perfect matching with high probability.*

Proof. Run Algorithm 2 $c \log n$ times for some constant $c \geq 2$. The probability of not getting a perfect matching at all iterations is at most $\frac{1}{n^c}$. Hence, we obtain a perfect matching with high probability. □

Remark 9.11. In fact, we can easily reduce time complexity of Algorithm 2 from $O(m \cdot n^\omega)$ to $O(n \log n \cdot n^\omega)$. Instead of looking at just one edge in each step, we pick one vertex and consider all the edges incident to it. Suppose that a bipartite graph G has a perfect matching. Then we pick a vertex u and let $e_1, \ldots, e_\ell$ denote the edges incident to u. Consider $G[E - \{e_1, \ldots, e_k\}]$ where $k = \lfloor \frac{\ell}{2} \rfloor$. If $G[E - \{e_1, \ldots, e_k\}]$ does not contain a perfect matching, then we know that any perfect matching in G has an edge in $\{e_1, \ldots, e_k\}$. In this case, we investigate $G\left[E - \left\{e_1, \ldots, e_{\lfloor \frac{k}{2} \rfloor}\right\}\right]$. If not, there is a perfect matching in G containing an edge in $\{e_{k+1}, \ldots, e_\ell\}$. In this case, we consider

$G\left[E - \left\{e_1, \ldots, e_{k+\lceil \frac{k}{2} \rceil}\right\}\right]$. We can repeat this binary-search-type procedure, and we can find v adjacent to u such that a perfect matching in G contains edge $\{u, v\}$. Then we delete u and v from G, and look at the resulting graph. There are at most $O(\log(\det(u))) = O(\log n)$ iterations in this procedure. Since there are at most n vertices to investigate and each iteration takes $O(n^\omega)$ time, it takes $O(n \log n \cdot n^\omega)$ in total. The following algorithm summarizes the aforementioned procedure.

Algorithm 3 Refined-Find-PM(G, S)

if PM-tester(G, S) returns 'NO' **then**
  return G does not have a perfect matching
else
  let U be a vertex in G
  let $e_1, \ldots, e_\ell$ be the edges incident to u
  do binary search to find k such that PM-tester($G\left[E - \{e_1, \ldots, e_k\}\right]$, S) returns 'YES' and PM-tester($G\left[E - \{e_1, \ldots, e_{k+1}\}\right]$, S) returns 'NO'
  let v denote the other end of e_{k+1}
  let Find-PM($G[V - \{u, v\}]$, S) return M'
  return $M' + e$
end if

3.2 Non-bipartite matching

In this section, we explain an algorithm for finding a perfect matching in non-bipartite graphs using the *Tutte matrix* which is defined as follows.

Definition 9.12. Suppose we have a graph general graph $G = (V, E)$ such that $|V| = n$. Then the Tutte matrix $T(G)$ of G is the $n \times n$ skew-symmetric matrix[1] given by

$$T_{i,j} = \begin{cases} 0 & \text{if } (i,j) \notin E \text{ or } i = j \\ x_{i,j} & \text{if } (i,j) \in E \text{ and } i < j \\ -x_{j,i} & \text{if } (i,j) \in E \text{ and } i > j. \end{cases}$$

Example 9.13. Consider the graph in Figure 9.2

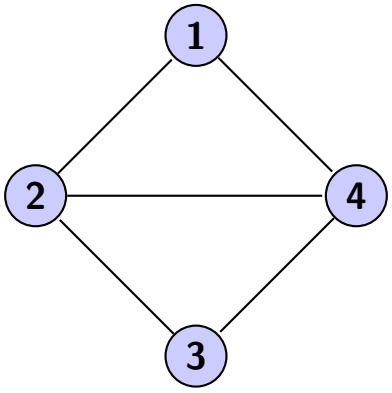

Figure 9.2: Non-bipartite graph

[1] A matrix A is said to be skew-symmetric if $A^T = -A$.

Notice that the corresponding Tutte matrix of this graph is the following.

$$\begin{bmatrix} 0 & x_{1,2} & 0 & x_{1,4} \\ -x_{1,2} & 0 & x_{2,3} & x_{2,4} \\ 0 & -x_{2,3} & 0 & x_{3,4} \\ -x_{1,4} & -x_{2,4} & -x_{3,4} & 0 \end{bmatrix}$$

An important property of this matrix is the following (we left the proof of the following theorem as an exercise).

Theorem 9.14. *The Tutte matrix $T(G)$ of G has non-zero determinant if and only if there exists a perfect matching in G.*

By Theorem 9.14, in order to check the existence of a perfect matching in a graph, it suffices to look at the determinant of its corresponding Tutte matrix. Besides, this implies that there exists an $O(m \cdot n^{\omega})$ time algorithm for perfect matching problem in non-bipartite graphs. We just need to slightly modify Algorithm 1. Observe that both Algorithm 2 and Algorithm 3 also work for non-bipartite graphs.

Corollary 9.15. *Given a graph G, not necessarily bipartite, there is an $\widetilde{O}(m \cdot n^{\omega})$ time algorithm which finds a perfect matching with high probability if one exists.*

Proof. We slightly modify Algorithm 1 to check if a given graph G contains a perfect matching. We construct the Tutte matrix of G, and compute its determinant. Then, it is easy to show that the resulting algorithm decides the existence of a perfect matching in G with probability at least $1 - \frac{1}{n^2}$ if $|S| \geq n^3$. If G contains a perfect matching, we use Algorithm 2 or Algorithm 3 to find one. □

4 $O(n \cdot n^{\omega})$ time algorithm by Rabin and Vazirani

In this section, we show an $O(n \cdot n^{\omega})$ algorithm for perfect matching problem due to Rabin and Vazirani [RV89]. Their algorithm works for both bipartite and non-bipartite graphs, but we are going to cover only the bipartite graph case in this note.

Let $G = (V, E)$ be a bipartite graph where $V = L \cup R$ and $|L| = |R| = n$. We denote by $\varepsilon(G)$ the Edmonds matrix of G. As we did in Section 3.1, we choose a value R_{ij} independently and uniformly at random from a set S of size at least n^3 for each non-zero entry ε_{ij} of $\varepsilon(G)$. Let $\widetilde{\varepsilon}$ denote the matrix after instantiating $\varepsilon(G)$ with those random values for its entries.

By Proposition 9.6, G has a perfect matching if and only if the determinant of $\varepsilon(G)$ is a non-zero polynomial. Therefore, $\det \widetilde{\varepsilon} \neq 0$ with probability at least $1 - \frac{1}{n^2}$. Moreover, we can find a permutation $\pi \in S_n$ such that $\widetilde{\varepsilon}_{i,\pi(i)} \neq 0$ for each $i \in [n]$ in this case. Rabin and Vazirani [RV89] proved that you can find such a permutation π in time $O(n \cdot n^{\omega})$.

If such a permutation $\pi \in S_n$ exists, we know that there is $j \in [n]$ such that $\widetilde{\varepsilon}_{1,j} \neq 0$ and $\det(\widetilde{\varepsilon}_{-1,-j}) \neq 0$. Based on this observation, we can construct the following algorithm.

Algorithm 4 Naive-RV-Bipartite(G, S)

construct the Edmonds matrix $\varepsilon(G)$ of a bipartite graph G
for each non-zero entry ε_{ij} in $\varepsilon(G)$, we sample a number $R_{ij} \in S$ independently and uniformly at random
instantiate $\varepsilon(G)$ with the chosen values for its entries and let's say it is $\widetilde{\varepsilon}$
Compute $\widetilde{\varepsilon}^{-1}$ and $\det(\widetilde{\varepsilon})$
if $\det(\widetilde{\varepsilon}) = 0$ **then**
    return G does not have a perfect matching
else
    for $j \in \{1, \ldots, n\}$ **do**
        if $\widetilde{\varepsilon}_{1,j} \neq 0$ and $\det(\widetilde{\varepsilon}_{-1,-j}) \neq 0$ **then**
            let u be the first vertex of L and v be the jth vertex of R
            let e be the edge with u and v as its two ends.
            return e + Naive-RV-Bipartite($G[V - \{u, v\}]$, S)
        end if
    end for
end if

Remark 9.16. Algorithm 4 runs in $O(n^2 \cdot n^\omega)$ time, because it involves computing determinants of n^2 submatrices of $\widetilde{\varepsilon}$. However, the algorithm given in the previous section provides better time complexity.

Therefore, we need to somehow reduce the number of determinant computation steps. In fact, we have an explicit formula for the inverse matrix of a given matrix by Cramer's rule:

$$\left(A^{-1}\right)_{i,j} = \frac{(-1)^{i+j} \det(A_{-j,-i})}{\det(A)}. \tag{9.1}$$

By (9.1), we get the following.

$$\det(\widetilde{\varepsilon}_{-1,-j}) = \left(\widetilde{\varepsilon}^{-1}\right)_{j,1} (-1)^{j+1} \det(\widetilde{\varepsilon}) \tag{9.2}$$

Once we compute the inverse and determinant of $\widetilde{\varepsilon}$, we can compute all of $\det(\widetilde{\varepsilon}_{-1,-1}), \ldots, \det(\widetilde{\varepsilon}_{-1,-n})$ using (9.1). Since we know that computing $\widetilde{\varepsilon}^{-1}$ and $\det(\widetilde{\varepsilon})$ takes $O(n^\omega)$ time by Bunch and Hopcroft [BH74], we can find $j \in [n]$ satisfying both $\widetilde{\varepsilon}_{1,j} \neq 0$ and $\det(\widetilde{\varepsilon}_{-1,-j}) \neq 0$ in time $O(n^\omega)$. This leads to the following refinement of Algorithm 4.

Algorithm 5 RV-Bipartite(G, S)

construct the Edmonds matrix $\varepsilon(G)$ of a bipartite graph G
for each non-zero entry ε_{ij} in $\varepsilon(G)$, we sample a number $R_{ij} \in S$ independently and uniformly at random
instantiate $\varepsilon(G)$ with the chosen values for its entries and let's say it is $\widetilde{\varepsilon}$
Compute $\widetilde{\varepsilon}^{-1}$ and $\det(\widetilde{\varepsilon})$
if $\det(\widetilde{\varepsilon}) = 0$ **then**
    return G does not have a perfect matching
else
    for $j \in \{1, \ldots, n\}$ **do**
        compute $\widetilde{\varepsilon}^{-1}$
        if $\widetilde{\varepsilon}_{1,j} \neq 0$ and $\det(\widetilde{\varepsilon}_{-1,-j}) \neq 0$ **then**
            let u be the first vertex of L and v be the j^{th} vertex of R
            let e be the edge with u and v as its two ends.
            return e + RV-Bipartite($G[V - \{u, v\}]$, S)
        end if
    end for
end if

Theorem 9.17. *Given a bipartite graph G, Algorithm 5 finds a perfect matching with probability at least $\frac{1}{2}$ if G contains one and it runs in time $O(n \cdot n^{\omega})$. Besides, there is an $\widetilde{O}(n \cdot n^{\omega})$ time algorithm which finds a perfect matching in G with high probability if one exists.*

Proof. Notice that we have replaced n^2 determinant computation steps with n steps, as desired. □

Remark 9.18. After we find $j \in [n]$ such that $\widetilde{\varepsilon}_{1,j} \neq 0$ and $\det(\widetilde{\varepsilon}_{-1,-j}) \neq 0$, we need to recurse on $\widetilde{\varepsilon}_{-1,-j}$. We remark that we can easily compute the inverse of $\widetilde{\varepsilon}_{-1,-j}$ once we have $\widetilde{\varepsilon}^{-1}$. This is due to the following formula: Let

$$A = \begin{bmatrix} a_{1,1} & v^T \\ u & B \end{bmatrix} \text{ and } A^{-1} = \begin{bmatrix} \widehat{a}_{1,1} & \widehat{v}^T \\ \widehat{u} & \widehat{B} \end{bmatrix} \tag{9.3}$$

where $\widehat{a}_{1,1} \neq 0$. Then

$$B^{-1} = \widehat{B} - \frac{1}{\widehat{a}_{1,1}} \widehat{u}\widehat{v}^T \tag{9.4}$$

5 Red-Blue perfect matching in bipartite graph

5.1 Red-Blue matching

Let us state first the *Red-Blue matching* problem.

- **Red-Blue matching problem:** We are given a graph G and a fixed number k. Then we color each of its edges in either red or blue. The problem is to find a perfect matching in G containing exactly k red edges.

In this section, we show how the polynomial method can be used to solve this problem. We assume that the input graph G is bipartite, and there exists a unique red-blue perfect matching. Let's

define a matrix M as follows.

$$M_{i,j} = \begin{cases} 0 & \text{if } (i,j) \notin E, \\ 1 & \text{if } (i,j) \in E \text{ and colored in blue,} \\ y & \text{if } (i,j) \in E \text{ and colored in red,} \end{cases}$$

Let $p(y)$ denote the determinant of M as a function of y. Then $p(y)$ is a polynomial in y with degree at most n. Observe that there exists a unique perfect matching with exactly k red edges if and only if $p(y)$ has a term of form $\pm y^k$. We can use Lagrangian interpolation method to find $p(y)$: choose $n+1$ distinct numbers $a_0, \ldots, a_n$, and evaluate $p(a_0) \ldots, p(a_n)$ by computing the determinant of M at $y = a_i$ for each i.

5.2 Isolation lemma

Let's consider a bipartite graph G. In the previous section, we look at the Red-Blue matching problem. One of the assumptions we made was the uniqueness of red-blue perfect matching contained in G. In this section, we will show that we can make G contain a unique minimum weight red-blue perfect matching with high probability.

G could potentially have $n!$ matchings. Now, we set edge weights randomly from a set of size $2m$, say $S = \{1, \ldots, 2m\}$. The following theorem is due to Mulmuley, Vazirani, and Vazirani [MVV87], and it is called "isolation lemma".

Theorem 9.19 (Mulmuley, Vazirani, and Vazirani [MVV87])**.** *The probability that there exists a unique minimum weight perfect matching is at least* $\frac{1}{2}$.

Proof. We call edge e "confused" if the weight of a minimum weight perfect matching containing e is the same as that of a minimum weight perfect matching not containing e. To prove this theorem, we need the following 2 claims.

Claim 9.20. *If M and M' are both minimum weight perfect matchings, then edge in $M \triangle M'$ is confused.*

Claim 9.21. *Let e be an edge in G. Then the probability that e is confused is at most* $\frac{1}{2m}$.

We first note that Claim 9.21 implies that the probability of the existence of a confused edge is at most $\frac{1}{2}$ by union bound. By Claim 9.20, the probability that there exist at least two minimum weight matchings is bounded above by the probability of the existence of a confused edge, which is at most $\frac{1}{2}$. Therefore, Claim 9.20 and Claim 9.21 imply that the probability that there exists a unique minimum weight perfect matching is at least $\frac{1}{2}$, as required.

To complete the proof, it suffices to prove both claims. Claim 9.20 is straightforward, so let us show Claim 9.21. Let e be an edge in G. We first set weights of all edges other than e. Let W_1 be the weight of a minimum weight perfect matching not containing e. Let W_2 be the minimum weight of a set S such that $S + e$ forms a perfect matching in G. If $W_2 \geq W_1$, then any perfect matching containing e has weight greater than W_1 and thus e is not confused. Hence, e is confused only if $W_2 < W_1$. If $W_2 < W_1$, e becomes confused when e has weight exactly $W_1 - W_2$. Therefore, the probability of e being confused is at most $\frac{1}{2m}$, as desired. □

Remark 9.22. Similarly, we can also show that the probability that there exists a unique minimum weight red-blue perfect matching is at least $\frac{1}{2}$. We just need to follow the proof of Theorem 9.19 and consider only red-blue perfect matchings.

Now, we define a matrix M as follows.

$$M_{i,j} = \begin{cases} 0 & \text{if } (i,j) \notin E, \\ 2^{w_{ij}} & \text{if } (i,j) \in E \text{ and colored in blue,} \\ 2^{w_{ij}} y & \text{if } (i,j) \in E \text{ and colored in red,} \end{cases}$$

where w_e denotes the randomly chosen weight of edge e. Let $p(y)$ denote the determinant of M as a function of y.

Corollary 9.23. *If G contains a red-blue perfect matching, then the coefficient of term y^k in $p(y)$ is non-zero with probability at least $\frac{1}{2}$.*

Proof. If is sufficient to show that if there is a unique minimum weight red-blue perfect matching in G, then the coefficient of y^k in $p(y)$ is non-zero. Let W denote the weight of the minimum weight red-blue perfect matching. Then, other red-blue perfect matchings have weight at least $W+1$. This implies the coefficient of term y^k can be written as $2^W + \sum_{i=1}^{\ell} (-1)^{a_i} 2^{W+b_i}$ where $a_i \in \{0,1\}$ and $b_i \geq 1$. Notice that

$$2^W + \sum_{i=1}^{\ell} (-1)^{a_i} 2^{W+b_i} = 2^W \left(1 + 2 \sum_{i=1}^{\ell} (-1)^{a_i} 2^{b_i - 1}\right).$$

Since $1 + 2\sum_{i=1}^{\ell} (-1)^{a_i} 2^{b_i-1}$ is an odd number, the coefficient of y^k cannot be zero. □

References

[BH74] James R. Bunch and John E. Hopcroft. Triangular factorization and inversion by fast matrix multiplication. *Mathematics of Computation*, 28(125):231–236, 1974. 9.8, 4

[DL78] Richard A. Demillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193 – 195, 1978. 9.2

[Lov79] László Lovász. On determinants, matchings, and random algorithms. In *FCT*, pages 565–574, 1979. 1, 3

[MS04] M. Mucha and P. Sankowski. Maximum matchings via gaussian elimination. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 248–255, Oct 2004. 1

[MV80] S. Micali and V. V. Vazirani. An $O(\sqrt{|V|}|E|)$ algoithm for finding maximum matching in general graphs. In *21st Annual Symposium on Foundations of Computer Science (sfcs 1980)*, pages 17–27, Oct 1980. 1

[MVV87] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987. 1, 5.2, 9.19

[RV89] M. O. Rabin and V. V. Vazirani. Maximum matchings in general graphs through randomization. *J. Algorithms*, 10(4):557–567, December 1989. 1, 4

[Sch80] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, October 1980. 9.2

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of the International Symposiumon on Symbolic and Algebraic Computation*, EUROSAM '79, pages 216–226, London, UK, UK, 1979. Springer-Verlag. 9.2