CS 252, Lecture 7: Matchings & Determinants

1 Introduction

In this lecture, we use algebraic tools to study perfect matchings in graphs. We will focus on bipartite graphs. Let $H = (L, R, E)$ denote a bipartite graph where L and R are the partitions of the vertex set and $E \subseteq L \times R$ is the set of the edges of the graph. Given such a graph H, our goal is to find if there is a perfect matching i.e. whether there is a bijection $\sigma : L \to R$ such that for every $\ell \in L$, $(\ell, \sigma(\ell)) \in E$. There is a standard augmenting path approach to this problem, but we view this from an algebraic perspective.

Let $|L| = |R| = n$. We represent H by the $n \times n$ adjacency matrix M as follows: the rows of M are indexed by the elements of L and the columns are indexed by elements of R, and

$$M_{\ell,r} = \begin{cases} 1 & \text{if } (\ell, r) \in E \\ 0 & \text{otherwise.} \end{cases}$$

2 Relation to Determinant

A question then arises: what kind of pattern in the above adjacency matrix corresponds to the existence of perfect matching in H?

This motivates the following definition:

$$P = \sum_{\sigma \in S_n} \prod_{\ell \in L} M_{\ell,\sigma(\ell)}$$

where S_n is the set of all the permutations of $\{1, 2, \ldots, n\}$. The above defined quantity is called as *permanent* of the matrix M. Note that in our case, it precisely counts the number of perfect matchings in H. However, computing the permanent of a matrix is a very hard problem, from a computational perspective. [1]

Thus, we look an alternative definition, the *determinant* of a square matrix M, denoted as $|M|$.

$$|M| = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{\ell \in L} M_{\ell,\sigma(\ell)}$$

[1] It is #P complete. #P is a class of counting problems for which a single solution is easy to verify, for eg. given a matching, it is easy to verify whether it is a perfect matching or not. However, counting the number of solutions can be very hard, just like counting the number of perfect matchings of a bipartite graph is hard.

Here, $\text{sgn}(\sigma)$, the signature or sign of a permutation is equal to $+1$, if there is an even number of swaps we can perform to σ to reach identity permutation, and -1 otherwise.

There are several other ways to define the determinant as well, for eg. we can define it recursively as follows: The determinant of a matrix $P = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ is equal to $ad - bc$. For the 3×3 matrix, it is defined as follows:

$$Q = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

$$|Q| = a \begin{vmatrix} \begin{pmatrix} e & f \\ h & i \end{pmatrix} \end{vmatrix} - b \begin{vmatrix} \begin{pmatrix} d & f \\ g & i \end{pmatrix} \end{vmatrix} + c \begin{vmatrix} \begin{pmatrix} d & e \\ g & h \end{pmatrix} \end{vmatrix}$$

Similarly, we can recursively define the determinant of a $n \times n$ matrix by using the definition of the determinant of a $(n-1) \times (n-1)$ matrix. Even though the signs seems rather arbitrary, determinant of matrices is very useful, and has a lot of applications. For eg., the determinant of a $n \times n$ square matrix is equal to the volume of the n dimensional parallelepiped spanned by the row (or column) vectors of the matrix. Thus, the determinant of a square matrix is non-zero if and only if the rows (or columns) of the matrix are linearly independent.

More importantly for us, a key distinction between the permanent and the determinant is that the determinant can indeed be computed very efficiently. We can use Gaussian elimination to transform the matrix into a lower triangular matrix, and then compute the determinant of the lower triangular matrix by simply multiplying the elements on the principal diagonal.

Thus, in order to check if H has a perfect matching or not, we can compute the determinant of the adjacency matrix M and check if it is equal to zero. Note that if there is no perfect matching in H, the determinant of M is equal to zero. However, the converse may not be true because of the signs involved in the definition of the determinant. It could be that there are several perfect matchings, and in computing the determinant with the signs, they could cancel each other out. One idea to fix this problem is to choose the elements of the adjacency matrix M cleverly. Note that as long as we assign non-edges to be zeroes, the determinant is zero if there is no perfect matching. Instead of assigning 1 to all the edges, we assign weights $W : E \to \{1, 2, \ldots, T\}$ and redefine the new adjacency matrix M as follows [2]:

$$M_{\ell,r} = \begin{cases} 2^{W_{(\ell,r)}} & \text{if } (\ell, r) \in E \\ 0 \text{ otherwise.} \end{cases}$$

Let the weight of a perfect matching $W(T) = \sum_{e \in E} W(e)$ be defined as the sum of the weights of its edges. The idea is to choose the weights cleverly such that one of the perfect matching stands out, thus making the determinant non-zero. In particular, we choose the weights W such that there is a unique perfect matching with minimum weight. We achieve this by choosing the weights W independently at random for each edge.

[2] We use the powers in 2 here to convert the products in determinant expression to sums of weights.

3 Isolation Lemma

The determinant of the redefined adjacency matrix is the following:

$$|M| = \sum_{\sigma} \begin{cases} 2^{\sum_{\ell \in L} W_{\ell,\sigma(\ell)}} & \text{if } \sigma \text{ is a perfect matching.} \\ 0 \text{ otherwise.} \end{cases} \tag{1}$$

Note that if there is no perfect matching, the value of $|M|$ is zero. We claim that if there is a unique perfect matching in H with minimum weight, then $|M|$ is non-zero.

Claim. *Let $W : E \to \{1, \ldots, T\}$ weights be assigned to the edges of the bipartite graph H, and M be as defined earlier. Suppose that there is a unique perfect matching in H with the least weight. Then, the determinant of M is non-zero.*

Proof. Let $W_{\min}$ denote the minimum weight of a perfect matching in H. As all the weights assigned to the edges are integers and the perfect matching with minimum weight is unique, there is exactly one perfect matching with weight $W_{\min}$ From Equation (1), we can write the determinant of M as $|M| = \pm 2^{W_{min}} + C\left(2^{W_{\min}+1}\right)$, for some constant C. This directly gives us that $|M| \neq 0$, as its remainder when divided by $2^{W_{\min}+1}$ is non-zero. □

As alluded to earlier, we can get this unique perfect matching with minimum weight property by randomly assigning weights to each edge. In fact, we can set $T = 2m$, where $m = |E|$. At first look, this is surprising since there could be exponentially many perfect matchings, and thus there would be a lot of collisions i.e. many perfect matchings will have the same weight irrespective of how we assign the weights. However, the least weight perfect matching is unique most of the time.

Lemma 1. *(Isolation Lemma)[[MVV87]] Let $\mathcal{F} \subseteq 2^{[m]}$ be a set family on a base set $E = [m] = \{1, 2, \ldots, m\}$. Let $W : E \to \{1, 2, \ldots, T\}$ be a weight function on E assigned uniformly and independently at random to each element of E. Then, with probability at least $1 - \frac{m}{T}$, there is a unique set in $\mathcal{F}$ with the smallest weight.*

Proof. We will outline proof from [Ta-15].

We will prove a stronger statement that there is a unique minimum weight set is at least $(1 - \frac{1}{T})^m$ probability over assigning the weights. First, we assume that no set in $\mathcal{F}$ is a subset of other. If such a set pair exists, we can remove the larger set as it will never be a minimum weight set. Let $\mathcal{W}$ denote the set of all weight assignments $W : E \to \{1, 2, \ldots, T\}$, and $\mathcal{W}_{>1}$ denote the set of weight assignments $W : E \to \{2, 3, \ldots, T\}$. We define a function $\phi : \mathcal{W}_{>1} \to \mathcal{W}$ as follows: For an assignment of weights $W \to \{2, 3, \ldots, T\}$, pick a set S_0 in $\mathcal{F}$ with minimum weight (as per W), and then let

$$W'(i) = \begin{cases} W(i) - 1 & \text{if } i \in S_0 \\ W(i) & \text{otherwise.} \end{cases}$$

We set $\phi(W) = W'$. We claim two things:

1. First, there is a unique set of minimum weight in $\mathcal{F}$ using the weights $\phi(W)$, for all $W \in \mathcal{W}_{>1}$. Note that S_0 is a set with minimum weight in W, and its weight reduced by $|S_0|$ from W to $\phi(W)$ and every other set's weight has reduced by at most $|S_0| - 1$ since we assumed that no set is a subset of other. Thus, we can infer that S_0 is the unique set with minimum weight as per the assignment of weights $\phi(W)$.

2. Second, for every $W_1 \neq W_2$ in $\mathcal{W}_{>0}$, $\phi(W_1) \neq \phi(W_2)$. If $\phi(W_1) = \phi(W_2)$, then according to the first claim, there is a unique set with minimum value in $\phi(W_1) = \phi(W_2)$. However, this implies that after adding 1 to the elements of that set, we get back both W_1 and W_2, contradicting the fact that $W_1 \neq W_2$.

Combining the two claims, we can infer that there are at least $|\mathcal{W}_{>1}|$ set of assignment of weights in $\mathcal{W}$ that ensure that there is a unique minimum weight set. The proof follows then by observing that $|\mathcal{W}_{>1}| = (T-1)^m$. □

Now, set $T = 2m$. Using the isolation lemma, we can conclude that after assigning the weights randomly, with at least 50% probability, there is a unique perfect matching. Combining with Section 3, this implies that the determinant of M is non-zero with at least 50% probability, if there is a perfect matching in H. As before, if there is no perfect matching, the determinant of M is zero. We can combine with the following algorithm to check if H has a perfect matching:

Algorithm to find if H has a perfect matching

1. Set weights $W : E \to \{1, 2, \ldots, 2m\}$ uniformly and independently at random.
2. Set the matrix M to be the adjacency matrix with weights 2^W.
3. Compute the determinant of M.
4. Return Yes iff the determinant is non-zero.

Note that the algorithm uses random assignment of the weights, and thus it could be wrong sometimes. In fact, the algorithm has an *one-sided* error. If H does not have a perfect matching, the algorithm always returns No. If H does have a perfect matching, the algorithm returns Yes with probability at least 50%. We can repeat the algorithm many times to reduce this error.

4 Polynomial Identity Testing

Instead of assignment weights at random, we can assign *indeterminates* to the adjacency matrix, and check if the resulting determinant polynomial is zero or not. To be more formal, we define the adjacency matrix M of the bipartite graph H as follows:

$$M_{\ell,r} = \begin{cases} x_{\ell,r} & \text{if } (\ell, r) \in E \\ 0 & \text{otherwise.} \end{cases}$$

We can observe that the determinant of M, when expanded as a polynomial, is equal to the zero polynomial if and only if H does not have a perfect matching. Thus, in order to check if H has a perfect matching or not, we can just check if this polynomial is equal to the zero polynomial or not. However there is a catch: this polynomial could have exponential number of terms! Thus, directly computing the coefficients of this polynomial is inefficient. We can instead evaluate the polynomial at an arbitrary point. Thus, given a polynomial as a "black box", can we efficiently check if it is equal to the zero polynomial? This problem is known as Polynomial Identity Testing (PIT). There are fast randomized algorithms to do this. [3] Finding deterministic algorithms for this problem is a very big open problem in theoretical computer science!

Thus, we get a different randomized algorithm to find if H has a perfect matching. However, this begs the question: why do all this when there are simple deterministic algorithms using augmenting paths? A major reason is that the deterministic augment path algorithm is inherently sequential. These randomized algorithms can be easily parallelized. Furthermore, they also generalize well to other settings such as weighted and non-bipartite graphs.

References

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

[Ta-15] Noam Ta-Shma. A simple proof of the isolation lemma. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:80, 2015.

[3] Randomized algorithms are algorithms that use random bits that may err sometimes, but succeed with good probability, like the one that we have seen in the previous section.