**15-859(M): Randomized Algorithms** **Lecturer:** Shuchi Chawla
**Topic:** Finding Perfect Matchings **Date:** 20 Sep, 2004
**Scribe:** Viswanath Nagarajan

## 3.1 The existence of perfect matchings in bipartite graphs

We will look at an efficient algorithm that determines whether a perfect matching exists in a given bipartite graph or not. This algorithm and its extension to finding perfect matchings is due to Mulmuley, Vazirani and Vazirani (1987). The algorithm is based on polynomial identity testing (see the last lecture for details on this).

A bipartite graph $G = (U, V, E)$ is specified by two disjoint sets $U$ and $V$ of vertices, and a set $E$ of edges between them. A *perfect matching* is a subset of the edge set $E$ such that every vertex has exactly one edge incident on it. Since we are interested in perfect matchings in the graph $G$, we shall assume that $|U| = |V| = n$. Let $U = \{u_1, u_2, \cdots, u_n\}$ and $V = \{v_1, v_2, \cdots, v_n\}$. The algorithm we study today has no error if $G$ does not have a perfect matching (no instance), and errs with probability at most $\frac{1}{2}$ if $G$ does have a perfect matching (yes instance). This is unlike the algorithms we saw in the previous lecture, which erred on no instances.

**Definition 3.1.1** *The Tutte matrix of bipartite graph $G = (U, V, E)$ is an $n \times n$ matrix $M$ with the entry at row $i$ and column $j$,*

$$M_{i,j} = \begin{cases} 0 & if(u_i, u_j) \notin E \\ x_{i,j} & if(u_i, u_j) \in E \end{cases}$$

The determinant of the Tutte matrix is useful in testing whether a graph has a perfect matching or not, as the following claim shows. Note that we do not think of this determinant as taking on some numeric value, but purely as a function of the variables $x_{i,j}$.

**Claim 3.1.2** *Det(M)$\neq$ 0 $\iff$ There exists a perfect matching in $G$*

**Proof:** We have the following expression for the determinant :

$$\text{Det}(M) = \sum_{\pi \in S_n} (-1)^{sgn(\pi)} \Pi_{i=1}^{n} M_{i,\pi(i)}$$

where $S_n$ is the set of all permutations on $[n]$, and $sgn(\pi)$ is the sign of the permutation $\pi$. There is a one to one correspondence between a permutation $\pi \in S_n$ and a (possible) perfect matching $\{(u_1, v_{\pi(1)}), (u_2, v_{\pi(2)}), \cdots, (u_n, v_{\pi(n)})\}$ in $G$. Note that if this perfect matching does not exist in $G$ (*i.e.* some edge $(u_i, v_{\pi(i)}) \notin E$) then the term corresponding to $\pi$ in the summation is 0. So we have

$$\text{Det}(M) = \sum_{\pi \in P} (-1)^{sgn(\pi)} \Pi_{i=1}^{n} x_{i,\pi(i)}$$

where $P$ is the set of perfect matchings in $G$. This is clearly zero if $P = \emptyset$, *i.e.*, if $G$ has no perfect matching. If $G$ has a perfect matching, there is a $\pi \in P$ and the term corresponding to $\pi$ is

$\Pi_{i=1}^{n} x_{i,\pi(i)} \neq 0$. Additionally, there is no other term in the summation that contains the same set of variables. Therefore, this term is not cancelled by any other term. So in this case, $\text{Det}(M) \neq 0$. ■

This claim gives us an easy way to test a bipartite graph for a perfect matching — we use the polynomial identity testing algorithm of the previous lecture on the Tutte matrix of $G$. We accept if the determinant is not identically 0, and reject otherwise. Note that $\text{Det}(M)$ has degree at most $n$. So we can test its identity on the field $Z_p$, where $p$ is a prime number larger than $2n$. From the analysis of the polynomial testing algorithm, we have the following :

- $G$ has no perfect matching $\implies$ Pr[accept]=0
- $G$ has a perfect matching $\implies$ Pr[accept]$\geq \frac{1}{2}$

The above algorithm shows that Perfect Matching for bipartite graphs is in RP. The general case is left as a homework exercise.

## 3.2 A parallel algorithm for finding perfect matchings

The algorithm for checking the existence of a perfect matching described in the previous section can be easily converted to one that actually computes a perfect matching as follows:

1. Pick $(u_i, v_j) \in E$.
2. Check if $G \backslash \{u_i, v_j\}$ has a perfect matching.
3. If YES, output $(u_i, v_j)$ to be in the matching and recurse on $G \backslash \{u_i, v_j\}$ (graph obtained after the removal of vertices $u_i$ and $v_j$).
4. If NO, recurse on $G - (u_i, v_j)$ (graph obtained after removing the edge $(u_i, v_j)$).

Note that this algorithm is inherently sequential in that we cannot speed up its running time considerably by using multple processors.

We now extend the algorithm of the previous section to one that runs efficiently on parallel processors. The model here is that there are polynomially many processors, and each is to run in parallel in poly-logarithmic time. It is known that there is an efficient parallel algorithm for computing the determinant of a matrix, and we will use this fact to obtain a parallel algorithm for finding a perfect matching.

We start with the following idea for a parallel algorithm : There is a processor for every edge $(u_i, v_j)$ that tests (in parallel) if edge $(u_i, v_j)$ is in some perfect matching or not. If this edge is in some perfect matching, the processor outputs $(u_i, v_j)$, else it outputs nothing.

We are immediately faced with the problem that there may be several perfect matchings in the graph, and the resulting output is not a matching. The algorithm may in fact return all the edges in the graph. So instead of testing whether an edge $(u_i, v_j)$ is in some perfect matching or not, we want to test whether an edge $(u_i, v_j)$ is in a specific perfect matching or not. The way we so