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 π. 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 π 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 π 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 this is to put random weights on the edges of the graph and test for the minimum weight perfect matching. We will see that the minimum weight perfect matching is unique with a good probability. This follows from the following *Isolation Lemma*.

Lemma 3.2.1 *Let $S = \{e_1, \cdots, e_m\}$ and $S_1, \cdots S_k \subseteq S$. For every element e_i there is a weight w_i picked u.a.r. from $\{0, 1, \cdots, 2m-1\}$. The weight of subset S_j is $w(S_j) = \sum_{e_i \in S_j} w_i$. Then, Pr[minimum weight set among $S_1, \cdots, S_k$ is unique] $\geq \frac{1}{2}$*

Proof: We will estimate the probability that the minimum weight set is not unique. Let us define an element e_i to be *tied* if

$$\min_{S_j | e_i \in S_j} w(S_j) = \min_{S_j | e_i \notin S_j} w(S_j)$$

It is easy to see that there exists a tied element if and only if the minimum weight subset is not unique. Below we bound the probability that a fixed element e_i is tied. The result will then follow using a union bound.

We use the principle of deferred decisions. Let us fix the weights $w_1, \cdots, w_m$ of all the elements except w_i. We want to bound $\Pr_{w_i}[e_i \text{ is tied } | w_1, \cdots, w_{i-1}, w_{i+1}, w_m]$. Let

$$W^- = \min_{S_j | e_i \notin S_j} w(S_j)$$

$$W^+ = \min_{S_j | e_i \in S_j} w(S_j)$$

with w_i assigned the value 0. It is easy to see that e_i is tied iff $W^- = W^+ + w_i$. So,
$\Pr_{w_i}[e_i \text{ is tied } | w_1, \cdots, w_{i-1}, w_{i+1}, w_m] = \Pr_{w_i}[w_i = W^- - W^+ \ | w_1, \cdots, w_{i-1}, w_{i+1}, w_m] \leq \frac{1}{2m}$.
The last inequality is because there is at most on value for w_i for which $W^- = W^+ + w_i$. This holds irrespective of the particular values of the other $w_{i'}$s. So $\Pr[e_i \text{ is tied}] \leq \frac{1}{2m}$, and

$$\Pr[\text{there exists a tied element}] \leq \sum_{i=1}^{m} \Pr[e_i \text{ is tied}] \leq \frac{1}{2}$$

Thus Pr[minimum weight set is unique] $\geq \frac{1}{2}$ ■

Now we can look at the parallel algorithm for finding a perfect matching. For each edge (u_i, v_j), we pick a random weight $w_{i,j}$, from $[2m-1]$, where $m = |E|$ is the number of edges in G. Let the sets S_j denote all the perfect matchings in G. Then the Isolation Lemma implies that there is a unique minimum weight perfect matching with at least a half probability. We assign the value $x_{i,j} = 2^{w_{i,j}}$ to the variables in the Tutte matrix M. Let D denote the resulting matrix. We use the determinant of D to determine the weight of the min-weight perfect matching, if it is unique, as suggested by the following lemma.

Lemma 3.2.2 *Let W_0 be the weight of the minimum weight perfect matching in G. Then,*

- *G has no perfect matching $\implies$ Det(D) $= 0$.*
- *G has a unique min-weight perfect matching $\implies$ Det(D) $\neq 0$ and the largest power of 2 dividing Det(D) is W_0.*

- *G has more than one min-weight perfect matching $\implies$ $Det(D) = 0$ or the largest power of 2 dividing $Det(D)$ is at least W_0.*

Proof: If G has no perfect matching, it is clear from claim 3.1.2 that $\mathrm{Det}(D) = 0$.

Now consider that case when G has a unique min-weight perfect matching. From the expression of the determinant, we have

$$\mathrm{Det}(D) = \sum_{\pi \in P} (-1)^{sgn(\pi)} \Pi_{i=1}^{n} 2^{w_{i,\pi(i)}} = \sum_{\pi \in P} (-1)^{sgn(\pi)} 2^{\sum_{i=1}^{n} w_{i,\pi(i)}} = \sum_{\pi \in P} (-1)^{sgn(\pi)} 2^{w(\pi)}$$

where $w(\pi)$ is the weight of the perfect matching corresponding to π and P is the set of all perfect matchings in G. Since there is exactly one perfect matching of weight W_0 and other perfect matchings have weight at least W_0+1, this evaluates to an expression of the form $\pm 2^{W_0} \pm 2^{W_0+1} \cdots \pm$ other powers of 2 larger than W_0. Clearly, this is non-zero, and the largest power of 2 dividing this is W_0.

Now consider the case when G has more than one min-weight perfect matchings. In this case, if the determinant is non-zero, every term in the sumation is a power of 2, at least 2^{W_0}. So 2^{W_0} divides $\mathrm{Det}(D)$. ∎

We refer to the submatrix of D obtained by removing the i-th row and j-th column by $D_{i,j}$. Note that this is a matrix corresponding to the bipartite graph $G \backslash \{u_i, v_j\}$. The parallel algorithm would run as follows.

1. Pick random weights $w_{i,j}$ for the edges of G.
2. Compute the weight W_0 of the min-weight perfect matching from $\mathrm{Det}(D)$ (using the parallel algorithm for computing the determinant).
3. If $\mathrm{Det}(D) = 0$, output "no perfect matching".
4. For each edge $(u_i, v_j) \in E$ do, in parallel,:
   (a) Evaluate $\mathrm{Det}(D_{i,j})$.
   (b) If $\mathrm{Det}(D_{i,j}) = 0$, output nothing.
   (c) Else, find the largest power of 2, $W_{i,j}$, dividing $\mathrm{Det}(D_{i,j})$.
   (d) If $W_{i,j} + w_{i,j} = W_0$, output (u_i, v_j).
   (e) Else, output nothing.

It is clear that, if G has no perfect matching, this algorithm returns the correct answer. Now suppose G has a unique minimum weight perfect matching, then from lemma 3.2.2, we see that precisely all the edges in the unique min-weight perfect matching are output. To see this, consider an edge (u_i, v_j) not in the unique min weight perfect matchi(P is the set of all perfect matchings in G)ng. From the lemma, $\mathrm{Det}(D_{i,j})$ is either zero (edge is not output in this case), or $W_{i,j}$ is at least as large as the min-weight perfect matching in $G \backslash \{u_i, v_j\}$. This in turn implies that $w_{i,j} + W_{i,j}$ is strictly larger than the min-weight perfect matching (W_0) of G, since the min-weight perfect matching is unique and does not contain edge (u_i, v_j). Therefore this edge is not output. Conversely, if an edge (u_i, v_j) is in the unique min-weight perfect matching, consider removing this edge from the matching, and we get a unique min-weight perfect matching in $G \backslash \{u_i, v_j\}$. So, in this case $W_{i,j} = W_0 - w_{i,j}$ and the edge is output.

Thus, we see that in case G does have a perfect matching, this algorithm outputs a correct perfect matching with probability at least $\frac{1}{2}$, which is the probability that we have a unique min-weight perfect matching (from the Isolation Lemma).