CMPUT 675: Randomized Algorithms **Fall 2005**

Lecture 18: Nov 8

Lecturer: Mohammad R. Salavatipour *Scribe: Jun Ma*

18.1 Perfect Mathings

Recall Schwartz-Zippel Theorem:

Theorem 18.1 *Let $Q(x_1, ..., x_n)$ be a multivariate polynomial of total degree d. Fix $S \subseteq F$ and supppose that $r_1, ..., r_n$ are chosen uniformly at random and independetly from S. Then*

$$\Pr[Q(r_1, ..., r_n) = 0 | Q(x_1, ..., x_n) \not\equiv 0] \leq \frac{d}{|S|}.$$

We can use this theorem for computing perfect matchings in a graph. Let $G(U \bigcup V, E)$ be a bipartite graph with $|U| = |V| = n$. $M \subseteq E$ is a perfect matching if in the subgraph of G induced by M every vertex has degree exactly one.

Definition 18.2 *The Tutte matrix of a bipartite graph $G(U \cup V, E)$ is an n by n matrix M, such that:*

$$M_{i,j} = \begin{cases} 0 & \text{if } u_i v_j \notin E, \\ x_{ij} & \text{otherwise} \end{cases}$$

Lemma 18.3 $\det(M) \neq 0 \iff G$ *has a perfect matching.*

Proof: By definition $\det(M) = \sum_{\Pi \in P_n} (-1)^{sgn(\Pi)} \prod_{i-1}^{n} M_{i,\Pi(i)}$, where P_n is the set of all permutation of $[n]$. Each term in the above summation corresponds to a potential perfect matching and the term is non-zero if and only if all the entries of the matrix corresponding to the edges of that potential perfect matching are non-zero. Since no two terms cancel each other out, the sum is non-zero iff G has a perfect matching. ■

So, using this lemma and Schwartz Zippel Theorem we can easily check for the existence of a perfect matching. We check if $det(M) = 0$ or not:

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

This suggests the following simple algorithm for finding a perfect matching:

Let F be $\mathbb{Z}_p$ for a prime $p \geq 2n$.

Algoirhtm for finding perfect matching:

- Pick $u_i v_j \in E$

- Check if $G - u_iv_j$ has a perfect matching
  - If yes then output u_iv_j and recursive on $G - u_iv_j$
  - If no then recursive on $G - u_iv_j$

Can we make this algorithm parallel? First we define some basic notaion for parallel algorithms.

Definition 18.4 *PRAM: is a model for computation in which we have P Synchronous processors running in parallel that have random access to a global memory.*

Once we have several processor accessing a shared memory we always have the problem of conflict. For example, if two processers want to write to a single memory location at the same time, or if one wants to write there and another one wants to read. There are some standard models for resolving conflict:

- EREW: exclusive read/exclusive write
- CREW: concurrent read/exclusive write
- CRCW: concurrent read/concurrent write

A common model that we also use here is CREW.

Definition 18.5 *NC is the set of languages that have a PRAM algorithm A s.t. $x \in \sum^*$*

- $x \in L \rightarrow A$ *accepts* x
- $x \notin L \rightarrow A$ *rejects* x
- p *(the number of proc's) is polynomial in* $|x|$
- *time is polylogarithmic in* $|x|$

RNC is the randomized version of NC, i.e. if $x \in L$ then A accepts with probability at least $1/2$ and if $x \notin L$ then A rejects x.

Definition 18.6 *Given a matrix M, a minor of M is the submatrix obtained from M by deleting a row and a column.*

It is known that computing $\det(M)$, a minor, and taking inverse of M can be computed in RNC.

Let's go back to the perfect matching problem and see if we can turn our sequential algorithm into a parallel version.

First idea: take a processor for every edge and check if u_iv_j is in a perfect matching in parallel. The problem with this idea is that there may be several perfect matchings and so the result is not a perfect matching (e.g. if G is a complete bipartite graph then all the edges are returned).

Second idea: Check for a specific perfect matching in G. How to do this? put weights on the edges such that the minimum perfect matching is unique. Then look for a minimum weight perfect matching To achieve this we put some random weights on the edges and show that with good probability, the minimum weight perfect matching is unique. Then we show how to modify the algorithm to look for a minimum weight perfect matching in parallel.

Lemma 18.7 (Isolation Lemma) *Let $X = \{x_1, ..., x_m\}$ be a set of elements and $F = \{s_1, ..., s_k\}$ is a family of subsets of X (all distinct). Let $w : X \to \{1...2m\}$ be a positive integer function chosen uniformly at random and independet for each element, then* $\Pr[$*min weight set in F is unique*$] \geq \frac{1}{2}$.

Remark: This may seem impossible at first glance. Note that k can be as big as 2^m. Since the weights of sets are in the range $\{1, \ldots, 2m^2\}$, we expect to see $\frac{2^m}{2m^2}$ sets of each value in $\{1...2m^2\}$!. **Proof:** Fix an item x_i and let:

- Y_i: is the set of all sets that contain x_i
- Z_i: is the set of all sets that don't contain x_i

Suppose that $w(x_i) = -\infty \to$. Then all min-value sets are among those in Y_i. Similarly, if $w(x_i) = +\infty \to$ then all min-value sets are among those in Z_i.

If we increase $w(x_i)$ from $-\infty$ to $+\infty$ there is exactly one value for which the min-value of Y_i becomes equal to min-value of Z_i. For that value of $w(x_i)$ we say x_i is ambiguous (we don't know if the min-value set has x_i or not). Other than that, we know exactly if x_i is in the min-value set or not. So $\Pr[x_i \text{ is ambiguous}] \leq \frac{1}{2m}$, which implies $\Pr[\exists \text{ an ambiguous } x_i] \leq m \cdot \frac{1}{2m} = \frac{1}{2}$. Thus:

$$\Pr[\text{no } x_i \text{ is ambiguous}] \geq \frac{1}{2}.$$

■

Let $x = E$ and F be the set of perfect matchings. The isolation lemma implies that if we set the weights randomly from $\{1, \ldots, 2m\}$, then with probability $\geq \frac{1}{2}$ the minimum weight perfect matching is unique. Let $x_{ij} = 2^{w_{ij}}$, w_{ij} is the weight of $u_i v_j$ and let B be the modified Tutte matrix with entries x_{ij} defined above.

Lemma 18.8 *If there is a unique minimum weight perfect matching in G and W is its weight then (i)*$\det(B) \neq 0$ *and (ii) the largest power of 2 dividing* $\det(B)$ *is* 2^W.

Proof: Other than the unique minimum weight perfect matching, all others have weights in $\{W+1, W+2, \ldots\}$. So the terms in $Det(B)$ are of the form $\{\pm 2^W, \pm 2^{W+1}, \ldots\}$. Take a factor 2^W out from all the terms. Since there is a unique term with value 2^W we will get a sum of of the form $2^W(1 + a_1 \cdot 2^1 + a_2 \cdot 2^2 + \ldots)$ where each a_i is an integer (possibly negative or zero). Since the sum of the terms in the paranthesis is odd it is non-zero and therefore $Det(B) \neq 0$, also 2^W is the largest power of 2 that divides it. ■

This suggests the following RNC algorithm for perfect matching. Let B_{ij} be the minor of B obtained by deleting row i and column j.

An *RNC* algorithm for perfect matching in bipartite graphs:

- Pick w_{ij} uniformly at random from $\{1...2m\}$, with $m = |E|$.
- Compute Tutte matrix B from w_{ij}'s and $\det(B)$ in parallel and compute the largest W s.t. 2^W divides $\det(B)$.
- If $\det(B) = 0 \to$ no perfect matching
- For each $u_i v_j \in E$ do in parallel
  - compute $\det(B_{ij})$ where B_{ij} is the minor of B by deleting row i and column j.
  - if $\det(B_{ij}) \neq 0$ find the largest power of 2, say $2^{W_{ij}}$ that divides $\det(B_{ij})$
  - if $W_{ij} + w_{ij} = W$ then output $u_i v_j$.